



```
PPPPPPPP      AAAAAA      SSSSSSSS      000000      P P P P P P P P      EEEEEEEEEEE      NN      NN      222222
PPPPPPPP      AAAAAA      SSSSSSSS      000000      P P P P P P P P      EEEEEEEEEEE      NN      NN      222222
PP      PP      AA      AA      SS      00      00      PP      PP      EE      NN      NN      22      22
PP      PP      AA      AA      SS      00      00      PP      PP      EE      NN      NN      22      22
PP      PP      AA      AA      SS      00      00      PP      PP      EE      NNNN      NN      22      22
PPPPPPPP      AA      AA      SSSSSS      00      00      P P P P P P P P      EEEEEEEEE      NN      NN      22
PPPPPPPP      AA      AA      SSSSSS      00      00      P P P P P P P P      EEEEEEEEE      NN      NN      22
PP      AAAAAAAAAA      SS      00      00      PP      EE      NN      NN      NN      22
PP      AAAAAAAAAA      SS      00      00      PP      EE      NN      NN      NN      22
PP      AA      AA      SS      00      00      PP      EE      NN      NN      NN      22
PP      AA      AA      SS      00      00      PP      EE      NN      NN      NN      22
PP      AA      AA      SSSSSSSS      000000      PP      EE      NN      NN      2222222222
PP      AA      AA      SSSSSSSS      000000      PP      EEEEEEEEEEE      NN      NN      2222222222
                                     ....
                                     ....
                                     ....
                                     ....

LL      I I I I I I      SSSSSSSS
LL      I I I I I I      SSSSSSSS
LL      I I      SS
LL      I I      SS
LL      I I      SS
LL      I I      SS
LL      I I      SSSSSS
LL      I I      SSSSSS
LL      I I      SS
LL      I I      SS
LL      I I      SS
LL      I I      SS
LL      I I      SS
LLLLLLLLLLLL      I I I I I I      SSSSSSSS
LLLLLLLLLLLL      I I I I I I      SSSSSSSS
```

```
1 0001 0 MODULE PASSOPEN2 ( %TITLE 'OPEN procedure'
2 0002 0 IDENT = '1-015' ! File: PASOPEN2.B32 Edit: SBL1015
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: Pascal Language Support
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains PASSOPEN2 and PASS$OPEN, which open a file.
36 0036 1
37 0037 1 ENVIRONMENT: User mode - AST reentrant
38 0038 1
39 0039 1 AUTHOR: Steven B. Lionel, CREATION DATE: 1-April-1981
40 0040 1
41 0041 1 MODIFIED BY:
42 0042 1
43 0043 1 1-001 - Original. SBL 1-April-1981
44 0044 1 1-002 - Give ACCMETINC message if ACCESS METHOD:=DIRECT and device
45 0045 1 is not a random-access device. Don't store PFV address in enable
46 0046 1 argument until the FCB pointer is stored. SBL 21-June-1982
47 0047 1 1-003 - In CHECK_KEY_XABS, don't assume that XAB_SUM is still linked
48 0048 1 to the FAB. SBL 8-July-1982
49 0049 1 1-004 - In CHECK_KEY_XABS, allow any defined, but unrecognized key type.
50 0050 1 Don't enable prompting on spooled terminals. SBL 20-July-1982
51 0051 1 1-005 - Copy DEV$V_RND to FCB$V_RND. SBL 29-July-1982
52 0052 1 1-006 - Store DEV$V_FOD instead of DEV$V_RND. Keep DEV$V_RND for
53 0053 1 ACCESS METHOD:=DIRECT test. SBL 29-July-1982
54 0054 1 1-007 - Call PASS$REMOVE FILE from OPEN_HANDLER to deallocate FCB and
55 0055 1 store STATUS. QAR FT3-002 SBL 30-Aug-1982
56 0056 1 1-008 - Use FAB$W_BLS on non-disk-or-terminal devices to determine file's
57 0057 1 recordsize. Correct RECORD_LENGTH check. Remove VMS V2 variant
```



```
.. 58      0058 1 | code. SBL 27-Sept-1982
.. 59      0059 1 | 1-009 - Remove special case for "resultant string overflow" error. Don't
.. 60      0060 1 | enable prompting for an input-only file. SBL 9-Dec-1982
.. 61      0061 1 | 1-010 - Add new bit FCB$V INITIATE_PROMPT. This is used instead of
.. 62      0062 1 | FCB$V PROMPT_ENABLE to determine whether a look-ahead of that file
.. 63      0063 1 | should initiate prompts on other files. This allows GETs from
.. 64      0064 1 | readonly files to initiate prompting on other files. SBL 15-Dec-1982
.. 65      0065 1 | 1-011 - If reopening for prompting fails because of RMS$_RAT, go back to
.. 66      0066 1 | no prompting. SBL 5-Jan-1983
.. 67      0067 1 | 1-012 - Allow "quiet reopen for READONLY" to occur for UNKNOWN.
.. 68      0068 1 | SBL 10-Jan-1983
.. 69      0069 1 | 1-013 - Remove PROMPT xxx keyword recognition, since this was never
.. 70      0070 1 | supported. Change method of recognizing duplicate keywords to one
.. 71      0071 1 | that is extensible past 32 codes. Add stream recordtype support
.. 72      0072 1 | in advance of compiler support. SBL 17-Aug-1983
.. 73      0073 1 | 1-014 - Make KEYWD NAME TABLE global for use by PAS$CLOSE2. SBL 19-Aug-1983
.. 74      0074 1 | 1-015 - Also set FCB$V OPI if SHARING:=READONLY and user wants write access
.. 75      0075 1 | to sequential file. SBL 24-Feb-1984
.. 76      0076 1 | --
.. 77      0077 1 |
```

```

79 0078 1 %SBTTL 'Declarations'
80 0079 1
81 0080 1 PROLOGUE DEFINITIONS:
82 0081 1
83 0082 1
84 0083 1 REQUIRE 'RTLIN:PASPROLOG';           ! Linkages, externals, PSECTs, structures
85 0147 1
86 0148 1 !+
87 0149 1 ! Linkage definitions for internal procedures.
88 0150 1 !-
89 0151 1
90 0152 1 LINKAGE
91 0153 1     CALL_FILL_KEY_XABS =
92 0154 1     CALL (REGISTER=6, REGISTER=7),
93 0155 1     CALL_CHECK_KEY_XABS =
94 0156 1     CALL (REGISTER=6, REGISTER=7);
95 0157 1
96 0158 1
97 0159 1 TABLE OF CONTENTS:
98 0160 1
99 0161 1
100 0162 1 FORWARD ROUTINE
101 0163 1     PASS$OPEN2: NOVALUE,           ! Called by compiled code
102 0164 1     PASS$OPEN: CALL_OPEN NOVALUE, ! Default OPEN called by RTL
103 0165 1     PASS$OPEN_IMPLICIT: JSB_OPEN_IMPLICIT NOVALUE, ! Open INPUT and OUTPUT
104 0166 1     FILL_KEY_XABS: CALL_FILL_KEY_XABS NOVALUE, ! Fill in KEY XABs
105 0167 1     CHECK_KEY_XABS: CALL_CHECK_KEY_XABS NOVALUE, ! Check KEY XABs
106 0168 1     OPEN_HANDLER,           ! Condition handler for PASS$OPEN
107 0169 1     EXIT_HANDLER: NOVALUE;   ! Established by PASS$OPEN
108 0170 1
109 0171 1
110 0172 1 MACROS:
111 0173 1
112 0174 1     NONE
113 0175 1
114 0176 1 EQUATED SYMBOLS:
115 0177 1
116 0178 1     NONE
117 0179 1
118 0180 1 FIELDS:
119 0181 1
120 0182 1     NONE
121 0183 1
122 0184 1 OWN STORAGE:
123 0185 1
124 0186 1
125 0187 1 !+
126 0188 1 ! Declare a longword which is used as a flag to indicate whether or not
127 0189 1 ! an exit handler has been declared.
128 0190 1 !-
129 0191 1
130 0192 1 OWN
131 0193 1     EXITH_DECLARED: INITIAL (0);
132 0194 1
133 0195 1 !+
134 0196 1 ! Declare two flags which indicate if the files INPUT and OUTPUT have ever
135 0197 1 ! been opened. If set, PASS$LOOK_AHEAD will not implicitly open them.
```

PAS\$OPEN2  
1-015

OPEN procedure  
Declarations

M 10  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 4  
(2)

```
: 136      0198 1  !-
: 137      0199 1
: 138      0200 1 GLOBAL
: 139      0201 1      PAS$GV_INPUT_OPENED: BYTE INITIAL (0),
: 140      0202 1      PAS$GV_OUTPUT_OPENED: BYTE INITIAL (0);
```



```
142 0203 1 %SBTTL 'PAS$OPEN2 - Open a file'
143 0204 1 GLOBAL ROUTINE PAS$OPEN2 (
144 0205 1 PFV: REF $PAS$PFV_FILE VARIABLE,
145 0206 1 KEYWORDS: VECTOR [, LONG]
146 0207 1 ): NOVALUE =
147 0208 1
148 0209 1 ++
149 0210 1 FUNCTIONAL DESCRIPTION:
150 0211 1
151 0212 1 This routine opens a Pascal file. This entry is called from the
152 0213 1 compiled code for the OPEN procedure.
153 0214 1
154 0215 1 CALLING SEQUENCE:
155 0216 1
156 0217 1 CALL PAS$OPEN2 (PFV.mr.r [, keyword.rz.v [, keyword.rz.v ...]])
157 0218 1
158 0219 1 FORMAL PARAMETERS:
159 0220 1
160 0221 1 PFV - The Pascal File Variable (PFV) passed by reference.
161 0222 1 The structure of the PFV is defined in PASPFV.REQ.
162 0223 1
163 0224 1 keyword - Zero or more values representing keywords in the
164 0225 1 OPEN procedure. Some keywords are followed by
165 0226 1 associated arguments and values. See PASOPNKEY.REQ
166 0227 1 for the keyword value definitions.
167 0228 1
168 0229 1 If the keyword ERROR is specified, it must be first
169 0230 1 in the list, otherwise it will be ignored.
170 0231 1
171 0232 1 IMPLICIT INPUTS:
172 0233 1
173 0234 1 NONE
174 0235 1
175 0236 1 IMPLICIT OUTPUTS:
176 0237 1
177 0238 1 NONE
178 0239 1
179 0240 1 COMPLETION STATUS:
180 0241 1
181 0242 1 NONE
182 0243 1
183 0244 1 SIDE EFFECTS:
184 0245 1
185 0246 1 See PAS$$OPEN.
186 0247 1
187 0248 1 SIGNALLED ERRORS:
188 0249 1
189 0250 1 See PAS$$OPEN.
190 0251 1
191 0252 1 --
192 0253 1
193 0254 2 BEGIN
194 0255 2
195 0256 2 LOCAL
196 0257 2 FCB: REF $PAS$FCB_CONTROL_BLOCK,
197 0258 2 PFV_ADDR: VOLATILE,
198 0259 2 UNWIND_ACT: VOLATILE,
```

```
! Open a file
! File variable
! Keywords and specifiers
```

```
! Control block
! Enable argument
! Enable argument
```

```
199 0260 2      ERROR_ADDR: VOLATILE;                ! Enable argument
200 0261 2
201 0262 2      BUILTIN
202 0263 2      ACTUALCOUNT,
203 0264 2      AP,
204 0265 2      CALLG;
205 0266 2
206 0267 2      ENABLE
207 0268 2      PASS$IO_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR);
208 0269 2
209 0270 2      !+
210 0271 2      ! Set PFV_ADDR enable argument.
211 0272 2      !-
212 0273 2
213 0274 2      PFV_ADDR = PFV [PFV$R_PFV];
214 0275 2
215 0276 2      !+
216 0277 2      ! If ERROR is present, it is the first keyword in the list. See
217 0278 2      ! if that is true. If so, set enable argument to the return address.
218 0279 2      !-
219 0280 2
220 0281 2      IF ACTUALCOUNT () GEQU 3      ! At least 3 needed for PFV, keyword, value
221 0282 2      THEN
222 0283 2          IF .KEYWORDS [0] EQL PASS$K_ERROR      ! Is it the ERROR keyword?
223 0284 2          THEN
224 0285 2              ERROR_ADDR = .KEYWORDS [1];
225 0286 2
226 0287 2      !+
227 0288 2      ! Validate and lock PFV.
228 0289 2      !-
229 0290 2
230 0291 2      PASS$VALIDATE_PFV (PFV [PFV$R_PFV]; FCB);
231 0292 2
232 0293 2      !+
233 0294 2      ! Set unwind action to unlock file.
234 0295 2      !-
235 0296 2
236 0297 2      UNWIND_ACT = PASS$K_UNWIND_UNLOCK;
237 0298 2
238 0299 2      !+
239 0300 2      ! If the file is already open, it's an error.
240 0301 2      !-
241 0302 2
242 0303 2      IF .PFV [PFV$V_OPEN]
243 0304 2      THEN
244 0305 2          $PASS$IO_ERROR (PASS$_FILALROPE,0);      ! File already open
245 0306 2
246 0307 2      !+
247 0308 2      ! Do the OPEN.
248 0309 2      !-
249 0310 2
250 0311 2      CALLG (.AP, PASS$OPEN);
251 0312 2
252 0313 2      !+
253 0314 2      ! Indicate successful completion
254 0315 2      ! Unlock the file variable.
255 0316 2      !-
```



PASSOPEN2  
1-015

OPEN procedure  
PASSOPEN2 - Open a file

C 11  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASSOPEN2.B32;1

Page 7  
(3)

```

: 256      0317 2
: 257      0318 2      FCB [FCB$L_STATUS] = 0;
: 258      0319 2      PFV [PFV$V_LOCK] = 0;
: 259      0320 2
: 260      0321 2      RETURN;
: 261      0322 2
: 262      0323 1      END;
```

! End of routine PASSOPEN2

.TITLE PASSOPEN2 OPEN procedure  
.IDENT \1-015\

.PSECT \_PASS\$DATA,NOEXE, PIC,2

00000000 00000 EXITH\_DECLARED:

```

      .LONG 0
00 00004 PASS$GV_INPUT_OPENED::
      .BYTE 0
00 00005 PASS$GV_OUTPUT_OPENED::
      .BYTE 0
```

.EXTRN PASSOPEN2, PASS\$OPEN  
.EXTRN PASS\$OPEN\_IMPLICIT  
.EXTRN PASS\$IO\_HANDLER  
.EXTRN PASS\$VALIDATE\_PFV  
.EXTRN PASS\$SIGNAL, PASSK\_FILALROPE

.PSECT \_PASS\$CODE,NOWRT, SHR, PIC,2

			00C4 00000	.ENTRY PASSOPEN2, Save R2,R6,R7	: 0204
	5E		08 C2 00002	SUBL2 #8, SP	
			7E D4 00005	CLRL ERROR_ADDR	: 0254
		04	AE 7C 00007	CLRG UNWIND_ACT	
	6D	0047	CF DE 0000A	MOVAL 3\$, (FP)	
	56	04	AC D0 0000F	MOVL PFV, R6	: 0274
08	AE		56 D0 00013	MOVL R6, PFV_ADDR	
	03		6C 91 00017	CMPB (AP), #3	: 0281
			0A 1F 0001A	BLSSU 1\$	
	19	08	AC D1 0001C	CMPL KEYWORDS, #25	: 0283
			04 12 00020	BNEQ 1\$	
	6E	0C	AC D0 00022	MOVL KEYWORDS+4, ERROR_ADDR	: 0285
		00000000G	00 16 00026 1\$:	JSB PASS\$VALIDATE_PFV	: 0291
	04	AE	01 D0 0002C	MOVL #1, UNWIND_ACT	: 0297
0E	07	A6	05 E1 00030	BBC #5, 7(R6), -2\$	: 0303
			7E D4 00035	CLRL -(SP)	: 0305
	7E	00G	8F 9A 00037	MOVZBL #PASSK_FILALROPE, -(SP)	
	00000000G	00	02 FB 0003B	CALLS #2, PASS\$SIGNAL	
			04 00042	RET	
	0000V	CF	6C FA 00043 2\$:	CALLG (AP), PASS\$OPEN	: 0311
		D4	A7 D4 00048	CLRL -44(FCB)	: 0318
	50	04	AC D0 0004B	MOVL PFV, R0	: 0319
	07	A0	80 8F 8A 0004F	BICB2 #128, 7(R0)	
			04 00054	RET	: 0323
			0000 00055 3\$:	.WORD Save nothing	: 0254
	50	08	AC D0 00057	MOVL 8(AP), R0	
	50	04	A0 D0 0005B	MOVL 4(R0), R0	
		F4	A0 9F 0005F	PUSHAB ERROR_ADDR	

PASSOPEN2  
1-015

OPEN procedure  
PASSOPEN2 - Open a file

D 11  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 8  
(3)

	FB	A0	9F	00062	PUSHAB	UNWIND_ACT
	FC	A0	9F	00065	PUSHAB	PFV_ADDR
		03	DD	00068	PUSHL	#3
		5E	DD	0006A	PUSHL	SP
00000000G	7E	04	AC	7D	MOVQ	4(AP), -(SP)
00	00		03	FB	CALLS	#3, PASS\$IO_HANDLER
			04	00077	RET	

: Routine Size: 120 bytes, Routine Base: \_PASS\$CODE + 0000

: 263 0324 1  
: 264 0325 1 !<BLF/PAGE>

```
266 0326 1 %SBTTL 'PAS$OPEN - Open a file'
267 0327 1 GLOBAL ROUTINE PAS$OPEN (
268 0328 1     IN PFV: REF $PAS$PFV FILE_VARIABLE,
269 0329 1     KEYWORDS: VECTOR [, [ONG]];
270 0330 1     FCB: REF $PAS$FCB CONTROL_BLOCK
271 0331 1 ): CALL_OPEN NOVALUE =
272 0332 1
273 0333 1 ++
274 0334 1 FUNCTIONAL DESCRIPTION:
275 0335 1
276 0336 1     This procedure opens the Pascal file specified by 'PFV'
277 0337 1     using the attributes possibly supplied as 'keyword'. PAS$OPEN
278 0338 1     is to be called from other RTL routines only after they have tested
279 0339 1     and set PFV$V_LOCK in 'PFV'.
280 0340 1
281 0341 1 CALLING SEQUENCE:
282 0342 1
283 0343 1     CALL PAS$OPEN (PFV.mr.r [, keyword.rz.v [, keyword.rz.v ...]])
284 0344 1
285 0345 1 FORMAL PARAMETERS:
286 0346 1
287 0347 1     PFV           - The Pascal File Variable (PFV) passed by reference.
288 0348 1                   The structure of the PFV is defined in PASPFV.REQ.
289 0349 1
290 0350 1     keyword       - Zero or more values representing keywords in the
291 0351 1                   OPEN procedure. Some keywords are followed by
292 0352 1                   associated arguments and values. See PASOPNKEY.REQ
293 0353 1                   for the keyword value definitions.
294 0354 1
295 0355 1 IMPLICIT INPUTS:
296 0356 1
297 0357 1     The PFV$V_LOCK bit is assumed to have been tested-and-set by the caller.
298 0358 1
299 0359 1 IMPLICIT OUTPUTS:
300 0360 1
301 0361 1     FCB           - The address of the File Control Block for the file
302 0362 1                   is returned in the register designated by the linkage.
303 0363 1
304 0364 1     PFV           - If the RELPFD or RELBUF bits are set in the PFV, those
305 0365 1                   self-relative addresses are resolved.
306 0366 1
307 0367 1 ROUTINE VALUE:
308 0368 1
309 0369 1     NONE
310 0370 1
311 0371 1 SIDE EFFECTS:
312 0372 1
313 0373 1     Declares an exit handler, if not already declared.
314 0374 1
315 0375 1     The file is opened. Internal RTL and RMS structures are allocated
316 0376 1     and initialized. The PFV is set to indicate an opened file.
317 0377 1
318 0378 1 SIGNALLED ERRORS:
319 0379 1
320 0380 1
321 0381 1
322 0382 1 --
```



```
323 0383 1
324 0384 2 BEGIN
325 0385 3
326 0386 4 LOCAL
327 0387 5 PFV: REF $PASSPFV_FILE_VARIABLE,      ! Pascal File Variable
328 0388 6 PFV_ADDR: VOLATILE,                  ! Address of PFV
329 0389 7 XABKEY_ADDR: VOLATILE,                ! Contains address of dynamically
330 0390 8                                     ! allocated KEY XABs.
331 0391 9 XABKEY_SIZE: VOLATILE,                ! Contains size in bytes of
332 0392 0                                     ! allocated KEY XABs.
333 0393 1
334 0394 2 XAB_FHC: $XABFHC_DECL,                ! RMS FHC XAB
335 0395 3 XAB_SUM: $XABSUM_DECL,                ! RMS SUM XAB
336 0396 4
337 0397 5 USER_ACTION_BPV: REF VECTOR [2, LONG], ! USER_ACTION routine descriptor
338 0398 6 USR_ORG: BYTE,                        ! User-specified organization
339 0399 7 USR_RFM: BYTE,                        ! User-specified record format
340 0400 8 USR_USZ: WORD,                        ! User-specified record size
341 0401 9 FILE_TYPE: SIGNED,                    ! Is this INPUT, OUTPUT or neither?
342 0402 0                                     ! Values can be:
343 0403 1                                     !     K_INPUT = -1
344 0404 2                                     !     K_NEITHER = 0
345 0405 3                                     !     K_OUTPUT = 1
346 0406 4 RESULT_NAME_STRING: VECTOR [MAXU(NAM$C_MAXRSS, LNM$C_NAMLENGTH), BYTE];
347 0407 5                                     ! Resultant name string
348 0408 6
349 0409 7 STACKLOCAL
350 0410 8 KEYWORDS_SEEN: BITVECTOR [PASSK_OPEKEYHI+1]; ! Keep track of keywords seen
351 0411 9
352 0412 0 !+
353 0413 1 PASS$AB_KEYWD_NAME_TABLE correlates the individual keyword code values to
354 0414 2 the names keyword. This is used to keep track of which keywords have
355 0415 3 been seen.
356 0416 4 !-
357 0417 5
358 0418 6 GLOBAL
359 0419 7 PASS$AB_KEYWD_NAME_TABLE: VECTOR [PASSK_OPEKEYHI+1, BYTE] PSECT (_PASSCODE)
360 0420 8 PRESET (
361 0421 9 [PASSK_FILE_NAME] = PASSK_FILE_NAME,
362 0422 0 [PASSK_DEFAULT_FILE_NAME] = PASSK_DEFAULT_FILE_NAME,
363 0423 1 [PASSK_HISTORY_OLD] = PASSK_HISTORY,
364 0424 2 [PASSK_HISTORY_NEW] = PASSK_HISTORY,
365 0425 3 [PASSK_HISTORY_UNKNOWN] = PASSK_HISTORY,
366 0426 4 [PASSK_HISTORY_READONLY] = PASSK_HISTORY,
367 0427 5 [PASSK_RECORD_LENGTH] = PASSK_RECORD_LENGTH,
368 0428 6 [PASSK_ACCESS_METHOD_SEQUENTIAL] = PASSK_ACCESS_METHOD,
369 0429 7 [PASSK_ACCESS_METHOD_DIRECT] = PASSK_ACCESS_METHOD,
370 0430 8 [PASSK_ACCESS_METHOD_KEYED] = PASSK_ACCESS_METHOD,
371 0431 9 [PASSK_RECORD_TYPE_FIXED] = PASSK_RECORD_TYPE,
372 0432 0 [PASSK_RECORD_TYPE_VARIABLE] = PASSK_RECORD_TYPE,
373 0433 1 [PASSK_RECORD_TYPE_STREAM] = PASSK_RECORD_TYPE,
374 0434 2 [PASSK_RECORD_TYPE_STREAM_CR] = PASSK_RECORD_TYPE,
375 0435 3 [PASSK_RECORD_TYPE_STREAM_LF] = PASSK_RECORD_TYPE,
376 0436 4 [PASSK_CARRIAGE_CONTROL_LIST] = PASSK_CARRIAGE_CONTROL,
377 0437 5 [PASSK_CARRIAGE_CONTROL_FORTAN] = PASSK_CARRIAGE_CONTROL,
378 0438 6 [PASSK_CARRIAGE_CONTROL_NONE] = PASSK_CARRIAGE_CONTROL,
379 0439 7 [PASSK_ORGANIZATION_SEQUENTIAL] = PASSK_ORGANIZATION,
```

```
380 0440 2 [PASSK_ORGANIZATION_RELATIVE] = PASSK_ORGANIZATION,  
381 0441 2 [PASSK_ORGANIZATION_INDEXED] = PASSK_ORGANIZATION,  
382 0442 2 [PASSK_DISPOSITION_SAVE] = PASSK_DISPOSITION,  
383 0443 2 [PASSK_DISPOSITION_DELETE] = PASSK_DISPOSITION,  
384 0444 2 [PASSK_DISPOSITION_PRINT] = PASSK_DISPOSITION,  
385 0445 2 [PASSK_DISPOSITION_PRINT_DELETE] = PASSK_DISPOSITION,  
386 0446 2 [PASSK_DISPOSITION_SUBMIT] = PASSK_DISPOSITION,  
387 0447 2 [PASSK_DISPOSITION_SUBMIT_DELETE] = PASSK_DISPOSITION,  
388 0448 2 [PASSK_ERROR] = PASSK_ERROR,  
389 0449 2 [PASSK_USER_ACTION] = PASSK_USER_ACTION,  
390 0450 2 [PASSK_SHARING_NONE] = PASSK_SHARING,  
391 0451 2 [PASSK_SHARING_READONLY] = PASSK_SHARING,  
392 0452 2 [PASSK_SHARING_READWRITE] = PASSK_SHARING);  
393 0453 2  
394 0454 2  
395 0455 2  
396 0456 2  
397 0457 2  
398 0458 2  
399 0459 2  
400 0460 2  
401 0461 2  
402 0462 2  
403 0463 2  
404 0464 2  
405 0465 2  
406 0466 2  
407 0467 2  
408 0468 2  
409 0469 2  
410 0470 2  
411 0471 2  
412 0472 2  
413 0473 2  
414 0474 2  
415 0475 2  
416 0476 2  
417 0477 2  
418 0478 2  
419 0479 2  
420 0480 2  
421 0481 2  
422 0482 2  
423 0483 2  
424 0484 2  
425 0485 2  
426 0486 2  
427 0487 2  
428 0488 2  
429 0489 2  
430 0490 2  
431 0491 2  
432 0492 2  
433 0493 2  
434 0494 2  
435 0495 2  
436 0496 2
```

## LITERAL

```
    K_INPUT = -1,  
    K_NEITHER = 0,  
    K_OUTPUT = 1;
```

```
! For FILE_TYPE - file is INPUT  
! For FILE_TYPE - neither INPUT nor OUTPUT  
! For FILE_TYPE - file is OUTPUT
```

## BUILTIN

```
    TESTBITCS,  
    TESTBITSC,  
    ACTUALCOUNT;
```

## BIND

```
    RAB = FCB: REF BLOCK [, BYTE],  
    FAB = FCB: REF $PASSFAB_FCB_STRUCT,  
    NAM = FCB: REF $PASSNAM_FCB_STRUCT;
```

```
!+ Establish local condition handler which will close the file upon  
!- an unwind.
```

```
ENABLE OPEN_HANDLER (PFV_ADDR, XABKEY_ADDR, XABKEY_SIZE);
```

```
!+ Fill in local XAB blocks.  
!-
```

```
$XABFHC_INIT (XAB=XAB_FHC, NXT=XAB_SUM);  
$XABSUM_INIT (XAB=XAB_SUM);
```

```
!+ Move PFV argument to local PFV.  
!-
```

```
PFV = .IN_PFV;
```

```
!+ If the PFD address is relative, resolve it.  
!-
```

```
IF .PFV [PFV$V_REL_PFD]  
THEN
```

```
    BEGIN  
    PFV [PFV$A_PFD] = .PFV [PFV$A_PFD] + PFV [PFV$R_PFV];
```

```
437 0497 3 PFV [PFV$V_RELPRD] = 0;
438 0498 END;
439 0499
440 0500
441 0501
442 0502
443 0503
444 0504 IF TESTBITSC (PFV [PFV$V_RELBUF])
445 0505 THEN
446 0506 PFV [PFV$A_BUFFER] = .PFV [PFV$A_BUFFER] + PFV [PFV$R_PFV];
447 0507
448 0508
449 0509
450 0510
451 0511
452 0512 IF TESTBITCS (EXITH_DECLARED)
453 0513 THEN
454 0514 BEGIN
455 0515
456 0516
457 0517
458 0518
459 0519 LOCAL
460 0520 EXITH_CONTROL_BLOCK: REF VECTOR [, LONG];
461 0521
462 0522 EXITH_CONTROL_BLOCK = PAS$GET_VM (PFV [PFV$R_PFV], 20); ! 5 longwords
463 0523 EXITH_CONTROL_BLOCK [1] = EXIT_HANDLER; ! Routine address
464 0524 EXITH_CONTROL_BLOCK [2] = 1; ! 1 additional longword
465 0525 EXITH_CONTROL_BLOCK [3] = EXITH_CONTROL_BLOCK [4]; ! Reason for exit
466 0526 $DCLEX (DESB[K=EXITH_CONTROL_BLOCK [0]]); ! Assume success
467 0527 END;
468 0528
469 0529
470 0530
471 0531
472 0532
473 0533 IF PFV [PFV$R_PFV] EQ LA PAS$FV_INPUT
474 0534 THEN
475 0535 FILE_TYPE = K_INPUT
476 0536 ELSE IF PFV [PFV$R_PFV] EQ LA PAS$FV_OUTPUT
477 0537 THEN
478 0538 FILE_TYPE = K_OUTPUT
479 0539 ELSE
480 0540 FILE_TYPE = K_NEITHER;
481 0541
482 0542
483 0543
484 0544
485 0545
486 0546
487 0547 FCB = PAS$GET_VM (PFV [PFV$R_PFV], PAS$K_FILE_DYN_BLN) + FCB$K_BLN;
488 0548
489 0549
490 0550
491 0551
492 0552
493 0553 RAB [RAB$B_BID] = RAB$C_BID;
```



```
494 0554 2 RAB [RAB$B_BLN] = RAB$C_BLN;  
495 0555 2 FAB [FAB$B_BID] = FAB$C_BID;  
496 0556 2 FAB [FAB$B_BLN] = FAB$C_BLN;  
497 0557 2 NAM [NAM$B_BID] = NAM$C_BID;  
498 0558 2 NAM [NAM$B_BLN] = NAM$C_BLN;  
499 0559 2 NAM [NAM$B_ESA] = RESULT_NAME_STRING;  
500 0560 2 NAM [NAM$B_ESS] = NAM$C_MAXRSS;  
501 0561 2 NAM [NAM$B_RSA] = RESULT_NAME_STRING;  
502 0562 2 NAM [NAM$B_RSS] = NAM$C_MAXRSS;  
503 0563 2 RAB [RAB$B_FAB] = FAB [0,0,0,0];  
504 0564 2 FAB [FAB$B_NAM] = NAM [0,0,0,0];  
505 0565 2 FAB [FAB$B_XAB] = XAB_FHC;  
506 0566  
507 0567  
508 0568 2+ Set initial values in FCB  
509 0569 2-  
510 0570  
511 0571 FCB [FCB$A_PFV] = PFV [PFV$R_PFV];  
512 0572  
513 0573 2+  
514 0574 2- Store FCB address in PFV and set the FCB_VALID bit.  
515 0575 2-  
516 0576  
517 0577 PFV [PFV$A_FCB] = FCB [FCB$R_FCB];  
518 0578 PFV [PFV$V_FCB_VALID] = 1;  
519 0579 PFV_ADDR = PFV [PFV$R_PFV];  
520 0580  
521 0581 2+  
522 0582 2- Get information from PFD that we need.  
523 0583 2-  
524 0584  
525 0585 BEGIN  
526 0586 LOCAL  
527 0587 PFD: REF $PASSPFD_FILE_DESCRIPTOR; ! Pascal File Descriptor  
528 0588  
529 0589 PFD = .PFV [PFV$A_PFD]; ! Get PFD address  
530 0590 FCB [FCB$W_ATTRIB] = .PFD [PFD$W_ATTRIB]; ! Set attributes  
531 0591 FCB [FCB$A_PFD] = PFD [PFD$R_PFD]; ! PFD address  
532 0592  
533 0593 2+  
534 0594 2- Set the default RECORD_LENGTH. If a TEXTFILE, it's 133. Otherwise,  
535 0595 2- its the length of the record (not including the length word for  
536 0596 2- a VARYING.)  
537 0597 2-  
538 0598  
539 0599 IF NOT .FCB [FCB$V_TEXT]  
540 0600 THEN  
541 0601 4 BEGIN  
542 0602 4 IF .PFD [PFD$L_LENGTH] GTRU 65535 ! Component type too long?  
543 0603 4 THEN  
544 0604 4 $PASSIO_ERROR (PASS_INVRECLN,1,.PFD [PFD$L_LENGTH]);  
545 0605 4 IF .FCB [FCB$V_VARYING]  
546 0606 4 THEN  
547 0607 4 RAB [RAB$W_USZ] = .PFD [PFD$L_LENGTH] - 2 ! Subtract for length word  
548 0608 4 ELSE  
549 0609 4 RAB [RAB$W_USZ] = .PFD [PFD$L_LENGTH];  
550 0610 4 END
```

PAS\$OPEN2  
1-015

OPEN procedure  
PAS\$OPEN - Open a file

J 11  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32:1

Page 14  
(4)

```
.. 551      0611      3      ELSE
.. 552      0612      3      RAB [RAB$W_USZ] = 133;      ! Textfile
.. 553      0613      3
.. 554      0614      3      USR_USZ = .RAB [RAB$W_USZ];
.. 555      0615      2      END;
.. 556      0616      2
```

```
558 0617 2 | +
559 0618 2 | Look through keyword list and set all specified attributes.
560 0619 2 | -
561 0620 2
562 0621 2 KEYWORDS_SEEN <0,32> = 0; ! Indicate no keywords seen
563 0622 2 KEYWORDS_SEEN <32,PAS$K_OPEKEYHI-31> = 0;
564 0623 2 INCR I FROM 0 TO (ACTUALCOUNT () - 2) DO
565 0624 2 BEGIN
566 0625 2
567 0626 2 LOCAL
568 0627 2 KEYWD_VALUE;
569 0628 2
570 0629 2 | +
571 0630 2 | Check for valid keyword code. If not already seen, use the
572 0631 2 | keyword.
573 0632 2 | -
574 0633 2
575 0634 2 KEYWD_VALUE = .KEYWORDS [I];
576 0635 2 IF .KEYWD_VALUE LSS PAS$K_OPEKEYLO OR .KEYWD_VALUE GTR PAS$K_OPEKEYHI
577 0636 2 THEN
578 0637 2 $PASSIO_ERROR (PAS$_INVARGPAS,0);
579 0638 2
580 0639 2 IF TESTBITS (KEYWORDS_SEEN [.PAS$AB_KEYWD_NAME_TABLE [.KEYWD_VALUE]])
581 0640 2 THEN
582 0641 2 CASE .KEYWD_VALUE FROM PAS$K_OPEKEYLO TO PAS$K_OPEKEYHI OF
583 0642 2 SET
584 0643 2
585 0644 2 [PAS$K_FILE_NAME]:
586 0645 2 BEGIN
587 0646 2 LOCAL
588 0647 2 FNS: WORD;
589 0648 2 FNS = .KEYWORDS [(I=.I+1)]; ! Get string size
590 0649 2 IF .FNS GTRU 255
591 0650 2 THEN
592 0651 2 $PASSIO_ERROR (PAS$_INVFILSYN,0);
593 0652 2 FAB [FAB$B_FNS] = .FNS;
594 0653 2 FAB [FAB$L_FNA] = .KEYWORDS [(I=.I+1)]; ! String address
595 0654 2 END;
596 0655 2
597 0656 2 [PAS$K_DEFAULT_FILE_NAME]:
598 0657 2 BEGIN
599 0658 2 LOCAL
600 0659 2 DNS: WORD;
601 0660 2 DNS = .KEYWORDS [(I=.I+1)]; ! Get string size
602 0661 2 IF .DNS GTRU 255
603 0662 2 THEN
604 0663 2 $PASSIO_ERROR (PAS$_INVFILSYN,0);
605 0664 2 FAB [FAB$B_DNS] = .DNS;
606 0665 2 FAB [FAB$L_DNA] = .KEYWORDS [(I=.I+1)]; ! String address
607 0666 2 END;
608 0667 2
609 0668 2 [PAS$K_HISTORY_OLD]:
610 0669 2 BEGIN
611 0670 2 FCB [FCB$V_OLD_FILE] = 1;
612 0671 2 FAB [FAB$V_GET] = 1;
613 0672 2 FAB [FAB$V_PUT] = 1;
614 0673 2 FAB [FAB$V_TRN] = 1;
```



```
615      0674      4      FAB [FAB$V_DEL] = 1;
616      0675      4      FAB [FAB$V_UPD] = 1;
617      0676      4      END;
618      0677      4
619      0678      4      [PAS$K_HISTORY_NEW]:
620      0679      4      BEGIN
621      0680      4      FAB [FAB$V_GET] = 1;
622      0681      4      FAB [FAB$V_PUT] = 1;
623      0682      4      FAB [FAB$V_TRN] = 1;
624      0683      4      FAB [FAB$V_DEL] = 1;
625      0684      4      FAB [FAB$V_UPD] = 1;
626      0685      4      END;
627      0686      4
628      0687      4      [PAS$K_HISTORY_UNKNOWN]:
629      0688      4      BEGIN
630      0689      4      FAB [FAB$V_CIF] = 1;
631      0690      4      FAB [FAB$V_GET] = 1;
632      0691      4      FAB [FAB$V_PUT] = 1;
633      0692      4      FAB [FAB$V_TRN] = 1;
634      0693      4      FAB [FAB$V_DEL] = 1;
635      0694      4      FAB [FAB$V_UPD] = 1;
636      0695      4      END;
637      0696      4
638      0697      4      [PAS$K_HISTORY_READONLY]:
639      0698      4      BEGIN
640      0699      4      FCB [FCB$V_OLD_FILE] = 1;      ! MUST be old
641      0700      4      FCB [FCB$V_READ_ONLY] = 1;
642      0701      4      FAB [FAB$V_GET] = 1;
643      0702      4      END;
644      0703      4
645      0704      4      [PAS$K_RECORD_LENGTH]:
646      0705      4      BEGIN
647      0706      4      IF .KEYWORDS [(I=.I+1)] GTRU 65535
648      0707      4      THEN
649      0708      4      $PAS$IO_ERROR (PAS$ INVRECLN,1,.KEYWORDS [I]);
650      0709      4      USR_USZ = .KEYWORDS [I];
651      0710      4
652      0711      4      !+
653      0712      4      | If RECORD_LENGTH specified for textfile, use it.
654      0713      4      | Otherwise, keep it so we can give an error later if
655      0714      4      | it doesn't match existing file's record length.
656      0715      4      |-
657      0716      4
658      0717      4      IF .FCB [FCB$V_TEXT]
659      0718      4      THEN
660      0719      4      RAB [RAB$W_USZ] = .USR_USZ;
661      0720      4      END;
662      0721      4
663      0722      4      [PAS$K_ACCESS_METHOD_SEQUENTIAL]:
664      0723      4      BEGIN
665      0724      4      FCB [FCB$V_SEQUENTIAL] = 1;
666      0725      4      FAB [FAB$V_SQO] = 1;      ! Optimize network access
667      0726      4      END;
668      0727      4
669      0728      4      [PAS$K_ACCESS_METHOD_DIRECT]:
670      0729      4      BEGIN
671      0730      4      FCB [FCB$V_DIRECT] = 1;
```

```
672      0731      4      RAB [RAB$L_KBF] = FCB [FCB$L_COMPONENT];
673      0732      4      RAB [RAB$V_UIF] = 1;      ! Update if record exists
674      0733      4      END;
675      0734      4
676      0735      4      [PAS$K_ACCESS_METHOD_KEYED]:
677      0736      4      BEGIN
678      0737      4      FCB [FCB$V_KEYED] = 1;
679      0738      4      END;
680      0739      4
681      0740      4      [PAS$K_RECORD_TYPE_FIXED]:
682      0741      4      BEGIN
683      0742      4      FAB [FAB$B_RFM] = FAB$C_FIX;
684      0743      4      END;
685      0744      4
686      0745      4      [PAS$K_RECORD_TYPE_VARIABLE]:
687      0746      4      BEGIN
688      0747      4      FAB [FAB$B_RFM] = FAB$C_VAR;
689      0748      4      END;
690      0749      4
691      0750      4      [PAS$K_RECORD_TYPE_STREAM]:
692      0751      4      BEGIN
693      0752      4      FAB [FAB$B_RFM] = FAB$C_STM;
694      0753      4      END;
695      0754      4
696      0755      4      [PAS$K_RECORD_TYPE_STREAM_CR]:
697      0756      4      BEGIN
698      0757      4      FAB [FAB$B_RFM] = FAB$C_STMCR;
699      0758      4      END;
700      0759      4
701      0760      4      [PAS$K_RECORD_TYPE_STREAM_LF]:
702      0761      4      BEGIN
703      0762      4      FAB [FAB$B_RFM] = FAB$C_STMLF;
704      0763      4      END;
705      0764      4
706      0765      4      [PAS$K_CARRIAGE_CONTROL_LIST]:
707      0766      4      BEGIN
708      0767      4      FAB [FAB$V_CR] = 1;
709      0768      4      END;
710      0769      4
711      0770      4      [PAS$K_CARRIAGE_CONTROL_FORTRAN]:
712      0771      4      BEGIN
713      0772      4      FAB [FAB$V_FTN] = 1;
714      0773      4      END;
715      0774      4
716      0775      4      [PAS$K_CARRIAGE_CONTROL_NONE]:
717      0776      4      ;      ! Do nothing
718      0777      4
719      0778      4      [PAS$K_ORGANIZATION_SEQUENTIAL]:
720      0779      4      BEGIN
721      0780      4      FAB [FAB$B_ORG] = FAB$C_SEQ;
722      0781      4      END;
723      0782      4
724      0783      4      [PAS$K_ORGANIZATION_RELATIVE]:
725      0784      4      BEGIN
726      0785      4      FAB [FAB$B_ORG] = FAB$C_REL;
727      0786      4      END;
728      0787      4
```

```
729 0788 3 [PAS$K_ORGANIZATION_INDEXED]:
730 0789 BEGIN
731 0790 FAB [FAB$B_ORG] = FAB$C_IDX;
732 0791 END;
733 0792
734 0793 [PAS$K_DISPOSITION_SAVE]:
735 0794 BEGIN
736 0795 FCB [FCB$V_SAVE] = 1;
737 0796 END;
738 0797
739 0798 [PAS$K_DISPOSITION_DELETE]:
740 0799 BEGIN
741 0800 FCB [FCB$V_DELETE] = 1;
742 0801 END;
743 0802
744 0803 [PAS$K_DISPOSITION_PRINT]:
745 0804 BEGIN
746 0805 FCB [FCB$V_SAVE] = 1;
747 0806 FCB [FCB$V_PRINT] = 1;
748 0807 END;
749 0808
750 0809 [PAS$K_DISPOSITION_PRINT_DELETE]:
751 0810 BEGIN
752 0811 FCB [FCB$V_PRINT] = 1;
753 0812 FCB [FCB$V_DELETE] = 1;
754 0813 END;
755 0814
756 0815 [PAS$K_DISPOSITION_SUBMIT]:
757 0816 BEGIN
758 0817 FCB [FCB$V_SUBMIT] = 1;
759 0818 FCB [FCB$V_SAVE] = 1;
760 0819 END;
761 0820
762 0821 [PAS$K_DISPOSITION_SUBMIT_DELETE]:
763 0822 BEGIN
764 0823 FCB [FCB$V_SUBMIT] = 1;
765 0824 FCB [FCB$V_DELETE] = 1;
766 0825 END;
767 0826
768 0827 [PAS$K_USER_ACTION]:
769 0828 BEGIN
770 0829 USER_ACTION BPV = .KEYWORDS [(I=.I+1)];
771 0830 FCB [FCB$V_USER_ACTION] = 1;
772 0831 END;
773 0832
774 0833 [PAS$K_SHARING_NONE]:
775 0834 BEGIN
776 0835 FAB [FAB$V_NIL] = 1;
777 0836 END;
778 0837
779 0838 [PAS$K_SHARING_READONLY]:
780 0839 BEGIN
781 0840 FAB [FAB$V_SHRGET] = 1;
782 0841 END;
783 0842
784 0843 [PAS$K_SHARING_READWRITE]:
785 0844 BEGIN
```



PASSOPEN2  
1-015

OPEN procedure  
PASS\$OPEN - Open a file

B 12  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 19  
(5)

```

: 786      0845      4      FAB [FAB$V_SHRPUT] = 1;
: 787      0846      4      FAB [FAB$V_SHRGET] = 1;
: 788      0847      4      FAB [FAB$V_SHRDEL] = 1;
: 789      0848      4      FAB [FAB$V_SHRUPD] = 1;
: 790      0849      3      END;
: 791      0850      3
: 792      0851      3      [PASSK_ERROR]:
: 793      0852      4      BEGIN
: 794      0853      4      |
: 795      0854      4      |   The ERROR parameter, if specified, was already processed
: 796      0855      4      |   in PASS$OPEN2, so just ignore it here.
: 797      0856      4      |
: 798      0857      4      |   = .I + 1; ! Ignore next parameter
: 799      0858      3      END;
: 800      0859      3
: 801      0860      3      [INRANGE,OUTRANGE]:
: 802      0861      3      $PASS$IO_ERROR (PASS$_INVARGPAS,0);    ! Invalid argument
: 803      0862      3
: 804      0863      3      TES;
: 805      0864      3
: 806      0865      2      END;
: 807      0866      2
```

```

809 0867 2      | +
810 0868 2      | - Apply defaults.
811 0869 2      |
812 0870 2      |
813 0871 2      | +
814 0872 2      | - Default FILE_NAME
815 0873 2      |
816 0874 2      |
817 0875 2      | IF NOT .KEYWORDS_SEEN [PAS$K_FILE_NAME]
818 0876 2      | THEN
819 0877 2      | BEGIN
820 0878 2      |
821 0879 2      | | +
822 0880 2      | | - If this is standard file INPUT or OUTPUT, attempt to translate
823 0881 2      | | PASS$INPUT or PASS$OUTPUT, respectively. If no translation, use
824 0882 2      | | SYSS$INPUT or SYSS$OUTPUT instead. If not INPUT or OUTPUT, use
825 0883 2      | | file variable name.
826 0884 2      | | -
827 0885 2      |
828 0886 2      | IF .FILE_TYPE NEQ K_NEITHER      ! neither INPUT nor OUTPUT?
829 0887 2      | THEN
830 0888 2      | BEGIN
831 0889 2      |
832 0890 2      | LOCAL
833 0891 2      | LOGNAM_DSC: BLOCK [8, BYTE],      ! Descriptor for logical name
834 0892 2      | RSLNAM_DSC: BLOCK [8, BYTE],      ! Descriptor for resultant name
835 0893 2      | SUBSTITUTE_NAME;                ! Address for substitute name
836 0894 2      |
837 0895 2      | IF .FILE_TYPE EQL K_INPUT
838 0896 2      | THEN
839 0897 2      | BEGIN
840 0898 2      | LOGNAM_DSC [DSC$W_LENGTH] = %CHARCOUNT ('PASS$INPUT');
841 0899 2      | LOGNAM_DSC [DSC$A_POINTER] = UPLIT BYTE ('PASS$INPUT');
842 0900 2      | SUBSTITUTE_NAME = UPLIT BYTE ('SYSS$INPUT');
843 0901 2      | END
844 0902 2      | ELSE
845 0903 2      | BEGIN
846 0904 2      | LOGNAM_DSC [DSC$W_LENGTH] = %CHARCOUNT ('PASS$OUTPUT');
847 0905 2      | LOGNAM_DSC [DSC$A_POINTER] = UPLIT BYTE ('PASS$OUTPUT');
848 0906 2      | SUBSTITUTE_NAME = UPLIT BYTE ('SYSS$OUTPUT');
849 0907 2      | END;
850 0908 2      | LOGNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
851 0909 2      | LOGNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
852 0910 2      | RSLNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
853 0911 2      | RSLNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
854 0912 2      | RSLNAM_DSC [DSC$W_LENGTH] = LNM$C_NAMLENGTH;      ! String size
855 0913 2      | RSLNAM_DSC [DSC$A_POINTER] = RESULT NAME STRING;
856 0914 2      | IF $TRNLOG (LOGNAM = LOGNAM_DSC, RSLNAM_DSC) EQLU SS$_NOTRAN
857 0915 2      | THEN
858 0916 2      | BEGIN
859 0917 2      | | +
860 0918 2      | | - Do the substitution
861 0919 2      | | -
862 0920 2      | | FAB [FAB$L_FNA] = .SUBSTITUTE_NAME;
863 0921 2      | | FAB [FAB$B_FNS] = .LOGNAM_DSC [DSC$W_LENGTH]; ! Can't be > 255
864 0922 2      | | END
865 0923 2      | ELSE
```

```
866      0924      5      BEGIN
867      0925      5      !+
868      0926      5      !- Use Pascal specific name
869      0927      5      !-
870      0928      5      FAB [FAB$L_FNA] = .LOGNAM_DSC [DSC$A_POINTER];
871      0929      5      FAB [FAB$B_FNS] = .LOGNAM_DSC [DSC$W_LENGTH]; ! Can't be >255
872      0930      4      END;
873      0931      3      END;
874      0932      3
875      0933      3      IF .FAB [FAB$L_FNA] EQL 0      ! No name given yet?
876      0934      3      THEN
877      0935      3          IF .FCB [FCB$V_EXTERN]      ! Filename is file variable name
878      0936      3          THEN
879      0937      4              BEGIN
880      0938      4              LOCAL
881      0939      4                  PFD: REF $PASS$PFD FILE_DESCRIPTOR;
882      0940      4                  PFD = .FCB [FCB$A_PFD];
883      0941      4                  FAB [FAB$B_FNS] = %CHRCHAR (PFD [PFD$T_NAME]);
884      0942      4                  FAB [FAB$L_FNA] = PFD [PFD$T_NAME] + 1;
885      0943      4                  END
886      0944      3          ELSE
887      0945      4              BEGIN
888      0946      4                  FAB [FAB$V_TMD] = 1;      ! No file name, temporary marked for delete
889      0947      4                  FAB [FAB$B_FNS] = %CHARCOUNT ('PASNONAME.TMP');
890      0948      4                  FAB [FAB$L_FNA] = UPLIT BYTE ('PASNONAME.TMP');
891      0949      4                  END;
892      0950      3      END;
893      0951      3
894      0952      3      !+
895      0953      3      !- Default DEFAULT_FILE_NAME
896      0954      3      !-
897      0955      3
898      0956      3      IF (NOT .KEYWORDS_SEEN [PASS$K_DEFAULT_FILE_NAME]) AND (.FCB [FCB$V_EXTERN])
899      0957      3      THEN
900      0958      3          BEGIN
901      0959      3              FAB [FAB$B_DNS] = %CHARCOUNT ('.DAT');
902      0960      3              FAB [FAB$L_DNA] = UPLIT BYTE ('.DAT');
903      0961      3          END;
904      0962      3
905      0963      3      !+
906      0964      3      !- Default HISTORY
907      0965      3      !-
908      0966      3
909      0967      3      IF NOT .KEYWORDS_SEEN [PASS$K_HISTORY]
910      0968      3      THEN
911      0969      3          BEGIN
912      0970      3              !+
913      0971      3              !- Default is NEW
914      0972      3              !-
915      0973      3              FAB [FAB$V_GET] = 1;
916      0974      3              FAB [FAB$V_PUT] = 1;
917      0975      3              FAB [FAB$V_TRN] = 1;
918      0976      3              FAB [FAB$V_DEL] = 1;
919      0977      3              FAB [FAB$V_UPD] = 1;
920      0978      3          END;
921      0979      3
922      0980      3      !+
```



```

923 0981 2  ! Default ACCESS_METHOD
924 0982 2  !-
925 0983 2  !-
926 0984 2  IF NOT .KEYWORDS_SEEN [PAS$K_ACCESS_METHOD]
927 0985 2  THEN
928 0986 2  BEGIN
929 0987 2  !-
930 0988 2  ! Default is SEQUENTIAL
931 0989 2  !-
932 0990 2  FCB [FCB$V_SEQUENTIAL] = 1;
933 0991 2  FAB [FAB$V_SQO] = 1; ! Optimize network access
934 0992 2  END
935 0993 2  ELSE IF .FCB [FCB$V_TEXT] AND NOT .FCB [FCB$V_SEQUENTIAL]
936 0994 2  THEN
937 0995 2  $PAS$IO_ERROR (PAS$TEXREQSEQ,0); ! Textfiles require sequential organization and access
938 0996 2  !-
939 0997 2  !+
940 0998 2  ! Default RECORD_TYPE
941 0999 2  !-
942 1000 2  !-
943 1001 2  IF NOT .KEYWORDS_SEEN [PAS$K_RECORD_TYPE]
944 1002 2  THEN
945 1003 2  IF .FCB [FCB$V_TEXT] OR .FCB [FCB$V_VARYING]
946 1004 2  THEN
947 1005 2  FAB [FAB$B_RFM] = FAB$C_VAR
948 1006 2  ELSE
949 1007 2  FAB [FAB$B_RFM] = FAB$C_FIX;
950 1008 2  USR_RFM = .FAB [FAB$B_RFM];
951 1009 2  !-
952 1010 2  !+
953 1011 2  ! Default CARRIAGE_CONTROL
954 1012 2  !-
955 1013 2  !-
956 1014 2  IF NOT .KEYWORDS_SEEN [PAS$K_CARRIAGE_CONTROL]
957 1015 2  THEN
958 1016 2  IF .FCB [FCB$V_TEXT] OR .FCB [FCB$V_VARYING]
959 1017 2  THEN
960 1018 2  FAB [FAB$V_CR] = 1; ! LIST is the default here
961 1019 2  !-
962 1020 2  !+
963 1021 2  ! Default DISPOSITION
964 1022 2  !-
965 1023 2  !-
966 1024 2  IF NOT .KEYWORDS_SEEN [PAS$K_DISPOSITION]
967 1025 2  THEN
968 1026 2  IF NOT .FAB [FAB$V_TMD] ! Not a temporary?
969 1027 2  THEN
970 1028 2  FCB [FCB$V_SAVE] = 1;
971 1029 2  !-
972 1030 2  !+
973 1031 2  ! Default ORGANIZATION.
974 1032 2  !-
975 1033 2  !-
976 1034 2  IF NOT .KEYWORDS_SEEN [PAS$K_ORGANIZATION]
977 1035 2  THEN
978 1036 2  FAB [FAB$B_ORG] = FAB$C_SEQ ! SEQUENTIAL
979 1037 2  ELSE IF .FCB [FCB$V_TEXT] AND (.FAB [FAB$B_ORG] NEQ FAB$C_SEQ)
```

PAS\$OPEN2  
1-015

OPEN procedure  
PAS\$OPEN - Open a file

F 12  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 23  
(6)

```

980      1038 2      THEN
981      1039      $PAS$IO_ERROR (PAS$ TEXREQSEQ,0);      ! Textfiles require sequential organization and access
982      1040      USR_ORG = .FAB [FAB$B_ORG];
983      1041
984      1042      !+
985      1043      ! Default SHARING
986      1044      !-
987      1045
988      1046      !+
989      1047      ! If ORGANIZATION:=SEQUENTIAL (explicitly), and if the file is shared
990      1048      ! so that there is at least one writer, then set FAB$V_UPI.
991      1049      !-
992      1050
993      1051      IF (.KEYWORDS_SEEN [PAS$K_ORGANIZATION] AND
994      1052      (.FAB [FAB$B_ORG] EQL FAB$C_SEQ))
995      1053      THEN
996      1054      IF .FAB [FAB$V_SHRPUT] OR
997      1055      (.FAB [FAB$V_PUT] AND .FAB [FAB$V_SHRGET])
998      1056      THEN
999      1057      FAB [FAB$V_UPI] = 1;
1000     1058
1001     1059      IF NOT .KEYWORDS_SEEN [PAS$K_SHARING]
1002     1060      THEN
1003     1061      IF .FCB [FCB$V_READ_ONLY]
1004     1062      THEN
1005     1063      FAB [FAB$V_SHRGET] = 1
1006     1064      ELSE
1007     1065      FAB [FAB$V_NIL] = 1;
1008     1066 2
```

```
1010 1067 2  !+
1011 1068 2  !- Check for conflicts
1012 1069 2  !+
1013 1070 2  !-
1014 1071 2  !+
1015 1072 2  !- Check for incompatible access method
1016 1073 2  !-
1017 1074 2  !+
1018 1075 2  !- IF NOT .FCB [FCB$V_OLD_FILE]
1019 1076 2  THEN
1020 1077 2  IF NOT (
1021 1078 2  (.FCB [FCB$V_SEQUENTIAL]) OR
1022 1079 2  (.FCB [FCB$V_DIRECT] AND
1023 1080 2  ((.FAB [FAB$B_ORG] EQL FAB$C_REL) OR
1024 1081 2  ((.FAB [FAB$B_ORG] EQL FAB$C_SEQ) AND (.FAB [FAB$B_RFM] EQL FAB$C_FIX))) OR
1025 1082 2  (.FCB [FCB$V_KEYED] AND
1026 1083 2  (.FAB [FAB$B_ORG] EQL FAB$C_IDX)))
1027 1084 2  THEN
1028 1085 2  $PASSIO_ERROR (PASS_ACCMETINC,0); ! Incompatible access method
1029 1086 2  !+
1030 1087 2  !- Check for incompatible DISPOSITION.
1031 1088 2  !-
1032 1089 2  !+
1033 1090 2  !- IF .FAB [FAB$V_TMD] AND
1034 1091 2  (.FCB [FCB$V_SAVE] OR .FCB [FCB$V_SUBMIT] OR .FCB [FCB$V_PRINT] OR
1035 1092 2  .FCB [FCB$V_OLD_FILE] OR .FAB [FAB$V_CIF])
1036 1093 2  THEN
1037 1094 2  $PASSIO_ERROR (PASS_FILNAMREQ,0); ! File name required
1038 1095 2  !+
1039 1096 2  !- Check for incompatible RECORD LENGTH. If creating non-text file
1040 1097 2  !- with fixed-length records, RECORD_LENGTH and component type must match.
1041 1098 2  !-
1042 1099 2  !+
1043 1100 2  !- IF NOT .FCB [FCB$V_TEXT] AND NOT .FCB [FCB$V_OLD_FILE] AND
1044 1101 2  (.FAB [FAB$B_RFM] EQL FAB$C_FIX) AND
1045 1102 2  (.USR_USZ NEQU .RAB [RAB$W_USZ])
1046 1103 2  THEN
1047 1104 2  $PASSIO_ERROR (PASS_RECLENINC,0);
1048 1105 2  !+
1049 1106 2  !- 
```



```
1051 1107 2      !+
1052 1108 2      ! If key descriptor block is present, allocate and fill in KEY XABs.
1053 1109 2      !-
1054 1110 2
1055 1111 2      BEGIN
1056 1112 2
1057 1113 2      LOCAL
1058 1114 2          PFD: REF $PASS$PFD_FILE_DESCRIPTOR; ! Pascal File Descriptor
1059 1115 2
1060 1116 2      PFD = .PFV [PFV$A_PFD];
1061 1117 2
1062 1118 2      IF .PFD [PFDSA_KDB] NEQ 0
1063 1119 2      THEN
1064 1120 2          FILL_KEY_XABS (PFV [PFV$R_PFV],
1065 1121 2                      FCB [FCB$R_FCB],
1066 1122 2                      PFD [PFDSR_PFD] + .PFD [PFDSA_KDB], ! Resolve self relative
1067 1123 2                      XAB_SUM,
1068 1124 2                      XABKEY_ADDR,
1069 1125 2                      XABKEY_SIZE);
1070 1126 2
1071 1127 2      END;
1072 1128 2
1073 1129 2      !+
1074 1130 2      ! Fill in remaining fields in RMS control blocks
1075 1131 2      !-
1076 1132 2
1077 1133 2      FAB [FAB$V_NEF] = 1;      ! Don't position to EOF
1078 1134 2      FAB [FAB$V_DFW] = 1;      ! Deferred write speeds up Relative and Indexed
1079 1135 2      RAB [RAB$V_RAH] = 1;      ! Read-ahead speeds up sequential
1080 1136 2      RAB [RAB$V_WBH] = 1;      ! Write-behind speeds up sequential
1081 1137 2
1082 1138 2      !+
1083 1139 2      ! Set maximum record size for all but variable-length sequential files,
1084 1140 2      ! unless the user specified RECORD_LENGTH.
1085 1141 2      !-
1086 1142 2
1087 1143 2      IF .FAB [FAB$B_ORG] NEQ FAB$C_SEQ OR
1088 1144 2          .FAB [FAB$B_RFM] EQL FAB$C_FIX OR
1089 1145 2          .KEYWORDS_SEEN [PASS$K_RECORD_LENGTH]
1090 1146 2      THEN
1091 1147 2          FAB [FAB$W_MRS] = .USR_USZ;
1092 1148 2
1093 1149 2      !+
1094 1150 2      ! If USER_ACTION was specified, call the user routine to do the
1095 1151 2      ! open and connect. Otherwise, do it here.
1096 1152 2      !-
1097 1153 2
1098 1154 2      BEGIN
1099 1155 2      LOCAL
1100 1156 2          STATUS;
1101 1157 2
1102 1158 2      IF .KEYWORDS_SEEN [PASS$K_USER_ACTION]
1103 1159 2      THEN
1104 1160 2          BEGIN
1105 1161 2          LINKAGE
1106 1162 2              USER_ACTION_LNK = CALL (REGISTER=1, STANDARD, STANDARD, STANDARD);
1107 1163 2          STATUS = USER_ACTION_LNK (.USER_ACTION_BPV [0], ! Address
```

```
1108 1164 4
1109 1165 4
1110 1166 4
1111 1167 4
1112 1168 4
1113 1169 3
1114 1170 4
1115 1171 4
1116 1172 4
1117 1173 4
1118 1174 4
1119 1175 4
1120 1176 5
1121 1177 6
1122 1178 6
1123 1179 6
1124 1180 7
1125 1181 8
1126 1182 8
1127 1183 9
1128 1184 8
1129 1185 7
1130 1186 6
1131 1187 6
1132 1188 4
1133 1189 4
1134 1190 4
1135 1191 4
1136 1192 4
1137 1193 4
1138 1194 4
1139 1195 4
1140 1196 4
1141 1197 4
1142 1198 4
1143 1199 4
1144 1200 4
1145 1201 5
1146 1202 5
1147 1203 5
1148 1204 5
1149 1205 5
1150 1206 5
1151 1207 5
1152 1208 5
1153 1209 5
1154 1210 6
1155 1211 7
1156 1212 7
1157 1213 7
1158 1214 8
1159 1215 7
1160 1216 7
1161 1217 5
1162 1218 4
1163 1219 4
1164 1220 4

      .USER_ACTION BPV [1],
      FAB [0,0,0,0],
      RAB [0,0,0,0],
      PFV [PFV$R_PFV]);
      ! Environment
      ! FAB address
      ! RAB address
      ! PFV address

      END
    ELSE
      BEGIN
        !+
        ! Open or create file.
        !-
        STATUS = (
          BEGIN
            LOCAL
              $$STATUS;
          DO ($$STATUS =
            (IF .FCB [FCB$V_OLD_FILE]
              THEN
                $OPEN (FAB=FAB [0,0,0,0])
              ELSE
                $CREATE (FAB=FAB [0,0,0,0]))
            UNTIL (.$$STATUS OR (.$$STATUS NEQU RMS$_ACT));
          .$$STATUS
          END);
        !+
        ! If we failed the open because we might not have been granted
        ! write permission to the file, try for just read permission.
        !-
        IF NOT .STATUS
        THEN
          IF (.STATUS EQLU RMS$_PRV) AND
            (.FCB [FCB$V_OLD_FILE] OR .FAB [FAB$V_CIF]) AND
            .FAB [FAB$V_PUT] ! PUT permission requested
          THEN
            BEGIN
              !+
              ! This was a $OPEN and we did not get permission to
              ! open the file. Set the FAC bits to just GET and
              ! try the $OPEN again. It might also fail.
              !-
              FAB [FAB$B_FAC] = FAB$_GET; ! Ask for only GET access
              FAB [FAB$V_NIL] = 0; ! Allow other sharing
              FAB [FAB$V_NAM] = 1; ! Use NAM block to reopen file
              STATUS = (
                BEGIN
                  LOCAL
                    $$STATUS;
                DO ($$STATUS = $OPEN (FAB=FAB [0,0,0,0]))
                UNTIL ($$STATUS NEQU RMS$_ACT);
                .$$STATUS
                END);
            END;
          !+

```

```
1165 1221 4      ! If open/create succeeded, enable prompting on this file if
1166 1222 4      ! it qualifies. Then connect the record stream.
1167 1223 4      !
1168 1224 4
1169 1225 4      IF .STATUS
1170 1226 4      THEN
1171 1227 5          BEGIN
1172 1228 5
1173 1229 5          !+
1174 1230 5          ! Set FCB$V_FOD if file is on a file oriented device. This is
1175 1231 5          ! used by $PASSRMS_OP to indicate that operations that fail with
1176 1232 5          ! RMSS_ACT can be retried.
1177 1233 5          !-
1178 1234 5
1179 1235 5          IF .BLOCK [FAB [FAB$L_DEV], DEV$V_FOD;4, BYTE]
1180 1236 5          THEN
1181 1237 5              FCB [FCB$V_FOD] = 1;
1182 1238 5
1183 1239 5          !+
1184 1240 5          ! If file is a terminal, and if prompting hasn't been
1185 1241 5          ! disabled, specify that look-ahead on this file will
1186 1242 5          ! initiate prompt output on PROMPT_ENABLED files.
1187 1243 5          !-
1188 1244 5
1189 1245 5          IF .FCB [FCB$V_TEXT] AND
1190 1246 5          .BLOCK [FAB [FAB$L_DEV], DEV$V_TRM;4, BYTE]
1191 1247 5          THEN
1192 1248 6              BEGIN
1193 1249 6                  FCB [FCB$V_INITIATE_PROMPT] = 1;
1194 1250 6
1195 1251 6          !+
1196 1252 6          ! We want to enable prompting on a file which is a terminal
1197 1253 6          ! and which has the LIST (CR) carriagecontrol attribute,
1198 1254 6          ! but which is not spooled and which can accept output.
1199 1255 6          !-
1200 1256 6
1201 1257 6          IF .FAB [FAB$V_CR] AND .FAB [FAB$V_PUT] AND
1202 1258 6          .BLOCK [FAB [FAB$L_DEV], DEV$V_ODV;4, BYTE] AND
1203 1259 6          NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_SPL;4, BYTE]
1204 1260 6          THEN
1205 1261 7              BEGIN
1206 1262 7                  $PASSRMS OP ($CLOSE (FAB=FAB [0,0,0,0]));
1207 1263 7                  FAB [FAB$B_RAT] = FAB$M_PRN;      ! Set print file format
1208 1264 7                  FAB [FAB$B_RFM] = FAB$C_VFC;      ! Set VFC format
1209 1265 7                  FAB [FAB$B_FSZ] = 2;              ! Set 2 byte control field
1210 1266 7                  FAB [FAB$V_NAM] = 1;              ! Use NAM block inputs
1211 1267 7                  STATUS = $PASSRMS_OP ($OPEN (FAB=FAB [0,0,0,0])); ! Reopen file
1212 1268 7
1213 1269 7          !+
1214 1270 7          ! If $OPEN failed, assume that we can't set up this
1215 1271 7          ! file for prompting. Therefore, go back to CR format.
1216 1272 7          !-
1217 1273 7
1218 1274 7          IF NOT .STATUS
1219 1275 7          THEN
1220 1276 8              BEGIN
1221 1277 8                  FAB [FAB$B_RAT] = FAB$M_CR;
```

PASSOPEN2  
1-015

OPEN procedure  
PASS\$OPEN - Open a file

K 12  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 28  
(8)

```

1222      FAB [FAB$B_RFM] = FAB$C_VAR;
1223      FAB [FAB$B_FSZ] = 0;
1224      P 1280      STATUS = $PASSRMS OP (
1225      1281      $OPEN (FAB=FAB [0,0,0,0]));
1226      1282      END
1227      1283      ELSE
1228      1284      BEGIN
1229      1285      !+
1230      1286      ! Open for prompting successful.
1231      1287      !-
1232      1288
1233      1289      FCB [FCB$V_PROMPT_ENABLE] = 1;
1234      1290      RAB [RAB$L_RHB] = FCB [FCB$W_PROMPT_CC]; ! Control area
1235      1291
1236      1292      END;
1237      1293      END;
1238      1294      END;
1239      1295      END;
1240      1296      END;
1241      1297      !+
1242      1298      ! If the OPEN/CREATE succeeded, do the CONNECT.
1243      1299      !-
1244      1300      IF .STATUS
1245      1301      THEN
1246      1302      DO (STATUS = $CONNECT (RAB=RAB [0,0,0,0]))
1247      1303      UNTIL (.STATUS OR (.STATUS NEQU RMSS_ACT));
1248      1304
1249      1305      END;
```



```
1251 1306 3      !+
1252 1307 3      !- If open/create or connect failed, signal an error
1253 1308 3      !-
1254 1309 3
1255 1310 3      IF NOT .STATUS
1256 1311 3      THEN
1257 1312 3          SELECTONE .STATUS OF
1258 1313 3              SET
1259 1314 3
1260 1315 3          [RMS$ FNF, RMS$ DNF, RMS$ DEV] :
1261 1316 3              $PASSIO_ERROR (PASS_FILNOTFOU);      ! File not found
1262 1317 3
1263 1318 3          [RMS$ FNM, RMS$ NOD, RMS$ DIR, RMS$ TYP, RMS$ VER, RMS$ SYN] :
1264 1319 3              $PASSIO_ERROR (PASS_INVFILSYN);      ! Invalid filename syntax
1265 1320 3
1266 1321 3          [OTHERWISE]:
1267 1322 3              $PASSIO_ERROR (PASS_ERRDUOPE);      ! Error during OPEN
1268 1323 3
1269 1324 3          TES;
1270 1325 3
1271 1326 3      END;
1272 1327 3
1273 1328 3      !+
1274 1329 3      !- If HISTORY := UNKNOWN has opened an existing file, indicate in the FCB
1275 1330 3      !-
1276 1331 3
1277 1332 3      IF .FAB [FAB$V_CIF] AND .FAB [FAB$L_STS] NEQU RMS$_CREATED
1278 1333 3      THEN
1279 1334 3          FCB [FCB$V_OLD_FILE] = 1;
1280 1335 3
1281 1336 3      !+
1282 1337 3      !- If we opened an existing file, get some attributes.
1283 1338 3      !-
1284 1339 3
1285 1340 3      IF .FCB [FCB$V_OLD_FILE]
1286 1341 3      THEN
1287 1342 3          BEGIN
1288 1343 3
1289 1344 3              !+
1290 1345 3              !- Check organization.
1291 1346 3              !-
1292 1347 3
1293 1348 3              IF .KEYWORDS_SEEN [PASS$K_ORGANIZATION] AND
1294 1349 3                  .FAB [FAB$B_ORG] NEQ .USR_ORG
1295 1350 3              THEN
1296 1351 3                  $PASSIO_ERROR (PASS_ORGSPEINC,0);      ! ORGANIZATION specified inconsistent
1297 1352 3
1298 1353 3              IF .FCB [FCB$V_TEXT] AND (.FAB [FAB$B_ORG] NEQ FAB$C_SEQ)
1299 1354 3              THEN
1300 1355 3                  $PASSIO_ERROR (PASS_TEXREQSEQ,0);      ! Textfiles require sequential organization and access
1301 1356 3
1302 1357 3              !+
1303 1358 3              !- Check recordtype.
1304 1359 3              !-
1305 1360 3
1306 1361 3              IF .KEYWORDS_SEEN [PASS$K_RECORD_TYPE]
1307 1362 3              THEN
```

```
1308 BEGIN
1309 IF .USR_RFM NEQ .FAB [FAB$B_RFM]
1310 THEN
1311 IF NOT ((.USR_RFM EQL FAB$C_VAR) AND
1312 (.FAB [FAB$B_RFM] EQL FAB$C_VFC))
1313 THEN
1314 $PASSIO_ERROR (PASS_RECTYPINC,0); ! Inconsistent record tyoe
1315 END;
1316
1317 !+
1318 ! Check record length.
1319 !-
1320
1321 !+
1322 ! If not a disk or terminal, use the blocksize as the maximum recordsize
1323 ! (if not there already).
1324 !-
1325 IF (NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_RND;4, BYTE]) AND
1326 (NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_TRM;4, BYTE])
1327 THEN
1328 IF .FAB [FAB$W_MRS] EQL 0
1329 THEN
1330 FAB [FAB$W_MRS] = .FAB [FAB$W_BLS];
1331
1332 IF .FAB [FAB$W_MRS] NEQ 0
1333 THEN
1334 IF .FAB [FAB$B_RFM] EQL FAB$C_FIX
1335 THEN
1336 BEGIN
1337 IF .FCB [FCB$V_TEXT]
1338 THEN
1339 BEGIN
1340 IF .KEYWORDS_SEEN [PASS$K_RECORD_LENGTH] AND
1341 (.USR_USZ NEQ .FAB [FAB$W_MRS])
1342 THEN
1343 $PASSIO_ERROR (PASS_RECLENINC,0);
1344 END
1345 ELSE IF (.RAB [RAB$W_USZ] NEQ .FAB [FAB$W_MRS]) OR
1346 (.USR_USZ NEQ .FAB [FAB$W_MRS])
1347 THEN
1348 $PASSIO_ERROR (PASS_RECLENINC,0);
1349 END
1350 ELSE
1351 BEGIN
1352 IF .FCB [FCB$V_TEXT] OR .FCB [FCB$V_VARYING]
1353 THEN
1354 BEGIN
1355 IF .KEYWORDS_SEEN [PASS$K_RECORD_LENGTH] AND
1356 (.USR_USZ GTRU .FAB [FAB$W_MRS])
1357 THEN
1358 $PASSIO_ERROR (PASS_RECLENINC,0);
1359 END
1360 ELSE IF (.RAB [RAB$W_USZ] GTRU .FAB [FAB$W_MRS]) OR
1361 (.USR_USZ GTRU .FAB [FAB$W_MRS])
1362 THEN
1363 $PASSIO_ERROR (PASS_RECLENINC,0);
1364 END;
```

```
1365 1420 3
1366 1421 3
1367 1422 3
1368 1423 3
1369 1424 3
1370 1425 3
1371 1426 3
1372 1427 3
1373 1428 3
1374 1429 3
1375 1430 3
1376 1431 4
1377 1432 4
1378 1433 4
1379 1434 4
1380 1435 3
1381 1436 3
1382 1437 3
1383 1438 3
1384 1439 3
1385 1440 3
1386 1441 4
1387 1442 4
1388 1443 5
1389 1444 6
1390 1445 4
1391 1446 5
1392 1447 4
1393 1448 3
1394 1449 3
1395 1450 3
1396 1451 2
1397 1452 2
1398 1453 2
1399 1454 2
1400 1455 2
1401 1456 2
1402 1457 2
1403 1458 2
1404 1459 2
1405 1460 2
1406 1461 3
1407 1462 2
1408 1463 2
1409 1464 2

      !+
      !- If textfile, get length from existing file.
      !-
      IF .FCB [FCBSV_TEXT]
      THEN
        IF .FAB [FAB$W_MRS] NEQ 0
        THEN
          RAB [RAB$W_USZ] = .FAB [FAB$W_MRS]
        ELSE
          BEGIN
            IF .XAB_FHC [XAB$W_LRL] GTRU .RAB [RAB$W_USZ]
            THEN
              RAB [RAB$W_USZ] = .XAB_FHC [XAB$W_LRL];
          END;

      !+
      !- Check for incompatible access method
      !-
      IF NOT (
        (.FCB [FCBSV_SEQUENTIAL]) OR
        (.FCB [FCBSV_DIRECT] AND
          ((.FAB [FAB$B_ORG] EQL FAB$C_REL) OR
           ((.FAB [FAB$B_ORG] EQL FAB$C_SEQ) AND (.FAB [FAB$B_RFM] EQL FAB$C_FIX)))) OR
        (.FCB [FCBSV_KEYED] AND
          (.FAB [FAB$B_ORG] EQL FAB$C_IDX)))
      THEN
        $PASSIO_ERROR (PASS$ACCMETINC,0); ! Incompatible access method
      END;

      !+
      !- For both old and new files, if direct access has been specified, see if
      !- device will allow it.
      !-
      IF .FCB [FCBSV_DIRECT]
      THEN
        IF (NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_RND;4, BYTE]) OR ! Random access?
        (.NAM [NAM$V_PPF]) ! Process-permanent file?
        THEN
          $PASSIO_ERROR (PASS$ACCMETINC,0);
```

```
1411 1465 2 1+
1412 1466 2 1- If the file is indexed organization, get information about the defined
1413 1467 2 keys.
1414 1468 2 1-
1415 1469 2
1416 1470 2 IF .FAB [FAB$B_ORG] EQL FAB$C_IDX
1417 1471 2 THEN
1418 1472 2 CHECK_KEY_XABS (PFV [PFV$R_PFV],
1419 1473 2 FCB [FCB$R_FCB],
1420 1474 2 XAB_SUM,
1421 1475 2 XABKEY_ADDR,
1422 1476 2 XABKEY_SIZE);
1423 1477 2
1424 1478 2 1+
1425 1479 2 1- If this is the implicit open of INPUT or OUTPUT, set the status bits to
1426 1480 2 1- cause an implicit RESET or REWRITE appropriately.
1427 1481 2
1428 1482 2
1429 1483 2 IF (.FILE_TYPE EQL K_INPUT) AND NOT .PAS$GV_INPUT_OPENED
1430 1484 2 THEN
1431 1485 2 BEGIN
1432 1486 2 PAS$GV_INPUT_OPENED = 1; 1 INPUT has been opened
1433 1487 2 PFV [PFV$V_VALID] = 0; 1 File variable not valid
1434 1488 2 FCB [FCB$V_LAZY] = 1; 1 Implicit GET on next access
1435 1489 2 FCB [FCB$V_INSPECTION] = 1; 1 In Inspection mode
1436 1490 2 END
1437 1491 2 ELSE IF (.FILE_TYPE EQL K_OUTPUT) AND NOT .PAS$GV_OUTPUT_OPENED
1438 1492 2 THEN
1439 1493 2 BEGIN
1440 1494 2 PAS$GV_OUTPUT_OPENED = 1; 1 OUTPUT has been opened
1441 1495 2 FCB [FCB$V_GENERATION] = 1; 1 In Generation mode
1442 1496 2 FCB [FCB$V_EOF] = 1; 1 At EOF
1443 1497 2 PFV [PFV$V_EOF_DEFINED] = 1; 1 EOF function defined
1444 1498 2 PFV [PFV$V_VALID] = 1; 1 File variable valid
1445 1499 2 PFV [PFV$V_DFB] = 0; 1 File buffer undefined
1446 1500 2 RAB [RAB$V_TPT] = 1; 1 Truncate on first $PUT
1447 1501 2 END
1448 1502 2 ELSE
1449 1503 2 BEGIN
1450 1504 2 1+
1451 1505 2 1- User must do RESET or REWRITE before using file.
1452 1506 2 1-
1453 1507 2 PFV [PFV$V_EOF_DEFINED] = 0; 1 EOF function undefined
1454 1508 2 PFV [PFV$V_VALID] = 1; 1 File variable valid
1455 1509 2 PFV [PFV$V_DFB] = 0; 1 File buffer undefined
1456 1510 2 END;
1457 1511 2
1458 1512 2 1+
1459 1513 2 1- Allocate the record buffer, if necessary, and fill in the UBF address.
1460 1514 2 1-
1461 1515 2
1462 1516 2 IF .FCB [FCB$V_TEXT]
1463 1517 2 THEN
1464 1518 2 BEGIN
1465 1519 2 RAB [RAB$L_UBF] = PAS$GET_VM (PFV [PFV$R_PFV], .RAB [RAB$W_USZ]);
1466 1520 2 FCB [FCB$V_DYNAMIC_UBF] = T;
1467 1521 2 END
```



```
1468 1522 2 ELSE IF .FCB [FCB$V_VARYING]
1469 1523 THEN
1470 1524 RAB [RAB$L_UBF] = .PFV [PFV$A_BUFFER] + 2
1471 1525 ELSE
1472 1526 RAB [RAB$L_UBF] = .PFV [PFV$A_BUFFER];
1473 1527
1474 1528 !+
1475 1529 ! Set up record pointers in FCB
1476 1530 !-
1477 1531
1478 1532 FCB [FCB$A_RECORD_BEG] = .RAB [RAB$L_UBF];
1479 1533 FCB [FCB$A_RECORD_CUR] = .RAB [RAB$L_UBF];
1480 1534 FCB [FCB$L_RECORD_LEN] = .RAB [RAB$W_USZ];
1481 1535 FCB [FCB$A_RECORD_END] = .FCB [FCB$A_RECORD_CUR] + .FCB [FCB$L_RECORD_LEN];
1482 1536
1483 1537 !+
1484 1538 ! If the file has no name (TMD) then zero the expanded and resultant
1485 1539 ! name strings.
1486 1540 !-
1487 1541
1488 1542 IF .FAB [FAB$V_TMD]
1489 1543 THEN
1490 1544 BEGIN
1491 1545 NAM [NAM$B_ESL] = 0;
1492 1546 NAM [NAM$B_RSL] = 0;
1493 1547 END
1494 1548
1495 1549 ELSE
1496 1550
1497 1551 !+
1498 1552 ! Allocate space for the resultant name string, move the local string
1499 1553 ! to the allocated one, and change the RSA pointer.
1500 1554 !-
1501 1555
1502 1556 BEGIN
1503 1557 NAM [NAM$L_RSA] = PAS$GET_VM (PFV [PFV$R_PFV], .NAM [NAM$B_RSL]);
1504 1558 CH$MOVE (.NAM [NAM$B_RSL], RESULT_NAME_STRING, .NAM [NAM$L_RSA]);
1505 1559 FCB [FCB$V_DYNAMIC_RSN] = 1; ! Indicate dynamic resultant name
1506 1560 END;
1507 1561
1508 1562 !+
1509 1563 ! Set linelimit.
1510 1564 !-
1511 1565
1512 1566 FCB [FCB$L_LINELIMIT] = -1; ! Initially indicate infinite limit
1513 1567
1514 1568 IF .FCB [FCB$V_TEXT] ! If textfile then look at PAS$LINELIMIT
1515 1569 THEN
1516 1570 BEGIN
1517 1571 LOCAL
1518 1572 LOGNAM_DSC: BLOCK [8, BYTE], ! Descriptor for logical name
1519 1573 LIMIT_DSC: BLOCK [8, BYTE]; ! Descriptor for limit string
1520 1574
1521 1575
1522 1576 !+
1523 1577 ! Translate the logical name PAS$LINELIMIT to find a possible
1524 1578 ! linelimit.
```

```
1525 1579 3      !-
1526 1580 3
1527 1581 3      LOGNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
1528 1582 3      LOGNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1529 1583 3      LOGNAM_DSC [DSC$W_LENGTH] = %CHARCOUNT ('PAS$LINELIMIT');
1530 1584 3      LOGNAM_DSC [DSC$A_POINTER] = UPLIT BYTE ('PAS$LINELIMIT');
1531 1585 3      LIMIT_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
1532 1586 3      LIMIT_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1533 1587 3      LIMIT_DSC [DSC$W_LENGTH] = LNM$C_NAMLENGTH;      ! Maximum resultant string length
1534 1588 3      LIMIT_DSC [DSC$A_POINTER] = RESULT_NAME_STRING;
1535 1589 3
1536 1590 3      IF $TRNLOG (LOGNAM = LOGNAM_DSC
P      1591 3          RSLLEN = LIMIT_DSC [DSC$W_LENGTH], ! Returned length
P      1592 4          RSLBUF = LIMIT_DSC)
1539 1593 3      EQLU SS$_NORMAL      ! Translation must succeed to continue
1540 1594 3      THEN
1541 1595 4      BEGIN
1542 1596 4      +
1543 1597 4      | Try to translate string as a signed decimal number. If
1544 1598 4      | successful, store as linelimit. Otherwise set linelimit
1545 1599 4      | back to -1.
1546 1600 4      |
1547 1601 4      -
1548 1602 4      IF NOT OT$SCVT_TI_L (LIMIT_DSC, FCB [FCB$L_LINELIMIT],
1549 1603 4          4,      ! Value size is 4 bytes
1550 1604 4          1)      ! Ignore blanks
1551 1605 4      THEN
1552 1606 4          FCB [FCB$L_LINELIMIT] = -1;
1553 1607 3      END;
1554 1608 3      END;
1555 1609 2
1556 1610 2      +
1557 1611 2      | Mark XAB chain invalid.
1558 1612 2      |
1559 1613 2      -
1560 1614 2      FAB [FAB$L_XAB] = 0;
1561 1615 2
1562 1616 2      +
1563 1617 2      | Deallocate any KEY XABs.
1564 1618 2      |
1565 1619 2      -
1566 1620 2      IF .XABKEY_ADDR NEQ 0
1567 1621 2      THEN
1568 1622 2          PAS$FREE_VM (.XABKEY_SIZE, XABKEY_ADDR);
1569 1623 2
1570 1624 2      +
1571 1625 2      | Mark PFV to indicate file open.
1572 1626 2      |
1573 1627 2      -
1574 1628 2      PFV [PFV$V_OPEN] = 1;
1575 1629 2
1576 1630 2      +
1577 1631 2      | Add file to list of open files.
1578 1632 2      |
1579 1633 2      -
1580 1634 2      PAS$ADD_FILE (FCB [FCB$R_FCB]);
1581 1635 2
```

```

: 1582      1636 2      RETURN;
: 1583      1637 2
: 1584      1638 1      END:

```

```
! End of routine PASS$OPEN
```

[illegible]

03	00	BE	1B	E5	00048	1\$:	BBCC	#27, @0(SP), 2\$	0504
		66	56	C0	0004D		ADDL2	PFV, (PFV)	0506
21	00000000	EF	00	E2	00050	2\$:	BBSS	#0, EXITH_DECLARED, 3\$	0512
			14	DD	00058		PUSHL	#20	0522
	00000000G	00	01	FB	0005A		CALLS	#1, PASS\$GET_VM	
	04	A0	CF	9E	00061		MOVAB	EXIT_HANDLER, 4(EXITH_CONTROL_BLOCK)	0523
	08	A0	01	D0	00067		MOVL	#1, 8(EXITH_CONTROL_BLOCK)	0524
	0C	A0	A0	9E	0006B		MOVAB	16(R0), 12(EXITH_CONTROL_BLOCK)	0525
			50	DD	00070		PUSHL	EXITH_CONTROL_BLOCK	0526
	00000000G	00	01	FB	00072		CALLS	#1, SYS\$DCLEXR	
	50	00000000G	00	9E	00079	3\$:	MOVAB	PASS\$FV INPUT, R0	0533
	50		56	D1	00080		CMPL	PFV, R0	
			06	12	00083		BNEQ	4\$	
	0C	AE	01	CE	00085		MNEGL	#1, FILE_TYPE	0535
			15	11	00089		BRB	6\$	
	50	00000000G	00	9E	0008B	4\$:	MOVAB	PASS\$FV OUTPUT, R0	0536
	50		56	D1	00092		CMPL	PFV, R0	
			06	12	00095		BNEQ	5\$	
	0C	AE	01	D0	00097		MOVL	#1, FILE_TYPE	0538
			03	11	0009B		BRB	6\$	
		0C	AE	D4	0009D	5\$:	CLRL	FILE_TYPE	0540
		0138	8F	3C	000A0	6\$:	MOVZWL	#312, -(SP)	0547
	00000000G	00	01	FB	000A5		CALLS	#1, PASS\$GET_VM	
	57	44	A0	9E	000AC		MOVAB	68(R0), FCB	
	67	4401	8F	B0	000B0		MOVW	#17409, (RAB)	0553
	5B	44	A7	9E	000B5		MOVAB	68(RAB), R11	0555
	6B		03	90	000B9		MOVW	#3, (R11)	
	45	A7	8F	90	000BC		MOVW	#80, 69(RAB)	0556
0094	C7	6002	8F	B0	000C1		MOVW	#24578, 148(RAB)	0557
00A0	C7	34	AE	9E	000C8		MOVAB	RESULT NAME STRING, 160(RAB)	0559
009E	C7		01	8E	000CE		MNEGB	#1, 158(RAB)	0560
04	AE	0098	C7	9E	000D3		MOVAB	152(RAB), 4(SP)	0561
04	BE	34	AE	9E	000D9		MOVAB	RESULT NAME STRING, @4(SP)	
0096	C7		01	8E	000DE		MNEGB	#1, 150(RAB)	0562
3C	A7		5B	D0	000E3		MOVL	R11, 60(RAB)	0563
6C	A7	0094	C7	9E	000E7		MOVAB	148(RAB), 108(RAB)	0564
68	A7	C8	AD	9E	000ED		MOVAB	XAB_FHC, 104(RAB)	0565
DC	A7		56	D0	000F2		MOVL	PFV, -36(RAB)	0571
0C	A6		57	D0	000F6		MOVL	RAB, 12(PFV)	0577
	1E		01	F0	000FA		INSV	#1, #30, #1, @0(SP)	0578
	FC		56	D0	00100		MOVL	PFV, PFV_ADDR	0579
	52	08	A6	D0	00104		MOVL	8(PFV), PFD	0589
	59	F8	A7	9E	00108		MOVAB	-8(RAB), R9	0590
	69	04	A2	B0	0010C		MOVW	4(PFD), (R9)	
	E4		52	D0	00110		MOVL	PFD, -28(RAB)	0591
	55	20	A7	9E	00114		MOVAB	32(RAB), R5	0607
	21		69	E8	00118		BLBS	(R9), 9\$	0599
	0000FFFF	8F	08	A2	D1	0011B	CMPL	8(PFD), #65535	0602
			06	1B	00123		BLEQU	7\$	
		08	A2	D0	00125		PUSHL	8(PFD)	0604
			01	31	00128		BRW	25\$	
07			02	E1	0012B	7\$:	BBC	#2, (R9), 8\$	0605
65	08	A2	02	A3	0012F		SUBW3	#2, 8(PFD), (R5)	0607
			0A	11	00134		BRB	10\$	
	65	08	A2	B0	00136	8\$:	MOVW	8(PFD), (R5)	0609
			04	11	0013A		BRB	10\$	0599
	65	85	8F	9B	0013C	9\$:	MOVZBW	#133, (R5)	0612



PASSOPEN2  
1-015

OPEN procedure  
PASS\$OPEN - Open a file

G 13  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 37  
(10)

		08	AE		65	B0	00140	10\$:	MOVW	(R5), USR_USZ	0614
				2C	AE	D4	00144		CLRL	KEYWORDS_SEEN	0621
		30	AE	01FF	8F	AA	00147		BICW2	#511, KEYWORDS_SEEN+4	0622
			58		6C	9A	0014D		MOVZBL	(AP), R8	0623
			58		02	C2	00150		SUBL2	#2, R8	
			54		01	CE	00153		MNEGL	#1, I	
					019F	31	00156	11\$:	BRW	59\$	
			53		08 AC44	D0	00159	12\$:	MOVL	KEYWORDS[I], KEYWD_VALUE	0634
					64	15	0015E		BLEQ	14\$	0635
			28		53	D1	00160		CMPL	KEYWD_VALUE, #40	
					5F	14	00163		BGTR	14\$	
			50		FE29 CF43	9A	00165		MOVZBL	PASS\$AB KEYWD_NAME_TABLE[KEYWD_VALUE], R0	0639
		2C	AE		50	E2	0016B		BBSS	R0, KEYWORDS_SEEN, -11\$	
			01		53	CF	00170		CASEL	KEYWD_VALUE, #1, #39	0641
								13\$:	.WORD	15\$-13\$,-	
00A3										16\$-13\$,-	
00E5										20\$-13\$,-	
010B										22\$-13\$,-	
012F										21\$-13\$,-	
015C										23\$-13\$,-	
0158										24\$-13\$,-	
0176										27\$-13\$,-	
0050										29\$-13\$,-	
0050										30\$-13\$,-	
011D										31\$-13\$,-	
										32\$-13\$,-	
										37\$-13\$,-	
										38\$-13\$,-	
										59\$-13\$,-	
										40\$-13\$,-	
										42\$-13\$,-	
										43\$-13\$,-	
										47\$-13\$,-	
										50\$-13\$,-	
										44\$-13\$,-	
										45\$-13\$,-	
										46\$-13\$,-	
										49\$-13\$,-	
										58\$-13\$,-	
										52\$-13\$,-	
										54\$-13\$,-	
										55\$-13\$,-	
										57\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										33\$-13\$,-	
										35\$-13\$,-	
										36\$-13\$,-	
										-(SP)	
		7E		00G	7E	D4	001C4	14\$:	CLRL	#PASSK_INVARGPAS, -(SP)	0861
					8F	9A	001C6		MOVZBL	18\$	
					30	11	001CA		BRB		

		54	D6	001CC	15\$:	INCL	I	0648		
	00FF	53	08	AC44	F7	001CE	CVTLW	KEYWORDS[I], FNS	0649	
		8F		53	B1	001D3	CMPW	FNS, #255	0652	
	78	A7		1C	1A	001D8	BGTRU	17\$	0653	
				53	90	001DA	MOVB	FNS, 120(RAB)	0641	
	70	A7		54	D6	001DE	INCL	I	0660	
				08	AC44	D0	001E0	MOVL	KEYWORDS[I], 112(RAB)	0661
				7A	11	001E6	BRB	28\$	0663	
				54	D6	001E8	INCL	I	0664	
	00FF	53		08	AC44	F7	001EA	CVTLW	KEYWORDS[I], DNS	0665
		8F		53	B1	001EF	CMPW	DNS, #255	0641	
				09	1B	001F4	BLEQU	19\$	0670	
	7E			7E	D4	001F6	CLRL	-(SP)	0680	
				00G	8F	9A	001F8	MOVZBL	#PASSK_INVFILSYN, -(SP)	0689
				05A2	31	001FC	BRW	147\$	0690	
	79	A7		53	90	001FF	MOVB	DNS, 121(RAB)	0694	
				54	D6	00203	INCL	I	0700	
	74	A7		08	AC44	D0	00205	MOVL	KEYWORDS[I], 116(RAB)	0701
				7C	11	0020B	BRB	34\$	0641	
	FC	A7		10	88	0020D	BISB2	#16, -4(RAB)	0670	
				04	11	00211	BRB	22\$	0680	
	4B	A7		02	88	00213	BISB2	#2, 75(RAB)	0689	
	50			5A	A7	9E	00217	MOVAB	90(RAB), R0	0690
	60			1F	88	0021B	BISB2	#31, (R0)	0694	
				0080	31	0021E	BRW	39\$	0641	
	FC	A7		18	88	00221	BISB2	#24, -4(RAB)	0700	
	5A	A7		02	88	00225	BISB2	#2, 90(RAB)	0701	
				7B	11	00229	BRB	41\$	0641	
				54	D6	0022B	INCL	I	0706	
	0000FFFF	8F		08	AC44	D1	0022D	CMP	KEYWORDS[I], #65535	0708
				12	1B	00236	BLEQU	26\$		
				08	AC44	DD	00238	PUSHL	KEYWORDS[I]	
				01	DD	0023C	PUSHL	#1		
		7E		00G	8F	9A	0023E	MOVZBL	#PASSK_INVRECLN, -(SP)	
	00000000G	00		03	FB	00242	CALLS	#3, PASS\$SIGNAL		
					04	00249	RET			
	08	AE		08	AC44	F7	0024A	CVTLW	KEYWORDS[I], USR_USZ	0709
		77		69	E9	00250	BLBC	(R9), 48\$	0717	
		65		08	AE	B0	00253	MOVW	USR_USZ, (R5)	0719
				7C	11	00257	BRB	51\$	0641	
	FC	A7		01	88	00259	BISB2	#1, -4(RAB)	0724	
	48	A7		40	8F	88	0025D	BISB2	#64, 72(RAB)	0725
					7E	11	00262	BRB	53\$	0641
	FC	A7		02	88	00264	BISB2	#2, -4(RAB)	0730	
	30	A7		D8	A7	9E	00268	MOVAB	-40(R7), 48(RAB)	0731
	04	A7		10	88	0026D	BISB2	#16, 4(RAB)	0732	
				7B	11	00271	BRB	56\$	0641	
	FC	A7		04	88	00273	BISB2	#4, -4(RAB)	0737	
				7F	11	00277	BRB	59\$	0641	
	63	A7		01	90	00279	MOVB	#1, 99(RAB)	0742	
				79	11	0027D	BRB	59\$	0641	
	63	A7		02	90	0027F	MOVB	#2, 99(RAB)	0747	
				73	11	00283	BRB	59\$	0641	
	63	A7		04	90	00285	MOVB	#4, 99(RAB)	0752	
				6D	11	00289	BRB	59\$	0641	
	63	A7		06	90	0028B	MOVB	#6, 99(RAB)	0757	
				67	11	0028F	BRB	59\$	0641	

63	A7		05	90	00291	36\$:	MOVB	#5, 99(RAB)	0762
			61	11	00295		BRB	59\$	0641
62	A7		02	88	00297	37\$:	BISB2	#2, 98(RAB)	0767
			5B	11	00298		BRB	59\$	0641
62	A7		01	88	0029D	38\$:	BISB2	#1, 98(RAB)	0772
			55	11	002A1	39\$:	BRB	59\$	0641
		61	A7	94	002A3	40\$:	CLRB	97(RAB)	0780
			50	11	002A6	41\$:	BRB	59\$	0641
61	A7		10	90	002A8	42\$:	MOVB	#16, 97(RAB)	0785
			4A	11	002AC		BRB	59\$	0641
61	A7		20	90	002AE	43\$:	MOVB	#32, 97(RAB)	0790
			44	11	002B2		BRB	59\$	0641
FC	A7	A0	8F	88	002B4	44\$:	BISB2	#160, -4(RAB)	0806
			3D	11	002B9		BRB	59\$	0641
FC	A7	80	8F	88	002BB	45\$:	BISB2	#128, -4(RAB)	0811
			0E	11	002C0		BRB	50\$	0812
FD	A7		01	88	002C2	46\$:	BISB2	#1, -3(RAB)	0817
FC	A7		20	88	002C6	47\$:	BISB2	#32, -4(RAB)	0818
			2C	11	002CA	48\$:	BRB	59\$	0641
FD	A7		01	88	002CC	49\$:	BISB2	#1, -3(RAB)	0823
FC	A7	40	8F	88	002D0	50\$:	BISB2	#64, -4(RAB)	0824
			21	11	002D5	51\$:	BRB	59\$	0641
			54	D6	002D7	52\$:	INCL	1	0829
	52	08	AC	44	D0	002D9	MOVL	KEYWORDS[I], USER_ACTION_BPV	
FD	A7		02	88	002DE		BISB2	#2, -3(RAB)	0830
			14	11	002E2	53\$:	BRB	59\$	0641
5B	A7		20	88	002E4	54\$:	BISB2	#32, 91(RAB)	0835
			0E	11	002E8		BRB	59\$	0641
5B	A7		02	88	002EA	55\$:	BISB2	#2, 91(RAB)	0840
			08	11	002EE	56\$:	BRB	59\$	0641
5B	A7		0F	88	002F0	57\$:	BISB2	#15, 91(RAB)	0848
			02	11	002F4		BRB	59\$	0641
			54	D6	002F6	58\$:	INCL	1	0857
FESB	54		58	F1	002F8	59\$:	ACBL	R8, #1, 1, 12\$	0623
03	2C	AE	01	E1	002FE		BBC	#1, KEYWORDS_SEEN, 60\$	0875
			0095	31	00303		BRW	67\$	
		0C	AE	D5	00306	60\$:	TSTL	FILE_TYPE	0886
			69	13	00309		BEQL	65\$	
FFFFFFFF	8F	0C	AE	D1	0030B		CMPL	FILE_TYPE, #-1	0895
			11	12	00313		BNEQ	61\$	
24	AE		09	B0	00315		MOVW	#9, LOGNAM_DSC	0898
28	AE	FC9F	CF	9E	00319		MOVAB	P.AAA, LOGNAM_DSC+4	0899
	53	FCA2	CF	9E	0031F		MOVAB	P.AAB, SUBSTITUTE_NAME	0900
			0F	11	00324		BRB	62\$	0895
24	AE		0A	B0	00326	61\$:	MOVW	#10, LOGNAM_DSC	0904
28	AE	FCA0	CF	9E	0032A		MOVAB	P.AAC, LOGNAM_DSC+4	0905
	53	FCA4	CF	9E	00330		MOVAB	P.AAD, SUBSTITUTE_NAME	0906
26	AE	010E	8F	B0	00335	62\$:	MOVW	#270, LOGNAM_DSC+2	0909
1C	AE	010E00FF	8F	D0	0033B		MOVL	#17694975, RSLNAM_DSC	0912
20	AE	34	AE	9E	00343		MOVAB	RESULT_NAME_STRING, RSLNAM_DSC+4	0913
			7E	7C	00348		CLRQ	-(SP)	0914
			7E	D4	0034A		CLRL	-(SP)	
		28	AE	9F	0034C		PUSHAB	RSLNAM_DSC	
			7E	D4	0034F		CLRL	-(SP)	
		38	AE	9F	00351		PUSHAB	LOGNAM_DSC	
00000000G	00		06	FB	00354		CALLS	#6, SYS\$TRNLOG	
00000629	8F		50	D1	0035B		CMPL	R0, #1577	

	70	A7		06	12	00362	BNEQ	63\$		
				53	D0	00364	MOVL	SUBSTITUTE_NAME, 112(RAB)	0920	
	70	A7	28	05	11	00368	BRB	64\$	0921	
	78	A7	24	AE	D0	0036A	63\$: MOVL	LOGNAM_DSC+4, 112(RAB)	0928	
			70	AE	90	0036F	64\$: MOVB	LOGNAM_DSC, 120(RAB)	0929	
				A7	D5	00374	65\$: TSTL	112(RAB)	0933	
10		69		22	12	00377	BNEQ	67\$		
		50		05	E1	00379	BBC	#5, (R9), 66\$	0935	
	78	A7	E4	A7	D0	0037D	MOVL	-28(RAB), PFD	0940	
	70	A7	OC	A0	90	00381	MOVB	12(PFD), 120(RAB)	0941	
			OD	A0	9E	00386	MOVAB	13(R0), 112(RAB)	0942	
	48	A7		OE	11	0038B	BRB	67\$	0935	
	78	A7		10	88	0038D	66\$: BISB2	#16, 72(RAB)	0946	
	70	A7		OD	90	00391	MOVB	#13, 120(RAB)	0947	
OE	2C	AE	FC49	CF	9E	00395	MOVAB	P.AAE, 112(RAB)	0948	
OA		69		02	E0	0039B	67\$: BBS	#2, KEYWORDS_SEEN, 68\$	0956	
	79	A7		05	E1	003A0	BBC	#5, (R9), 68\$		
	74	A7	FC43	04	90	003A4	MOVB	#4, 121(RAB)	0959	
07	2C	AE		CF	9E	003A8	MOVAB	P.AAF, 116(RAB)	0960	
		50	5A	03	E0	003AE	68\$: BBS	#3, KEYWORDS_SEEN, 69\$	0967	
		60		A7	9E	003B3	MOVAB	90(RAB), R0	0973	
	FC	OB	2D	1F	88	003B7	BISB2	#31, (R0)	0977	
	48	A7		AE	E8	003BA	69\$: BLBS	KEYWORDS_SEEN+1, 70\$	0984	
			40	01	88	003BE	BISB2	#1, -4(RAB)	0990	
				8F	88	003C2	BISB2	#64, 72(RAB)	0991	
		04		07	11	003C7	BRB	71\$	0984	
		4D	FC	69	E9	003C9	70\$: BLBC	(R9), 71\$	0993	
11	2D	AE		A7	E9	003CC	BLBC	-4(RAB), 79\$		
		04		03	E0	003D0	71\$: BBS	#3, KEYWORDS_SEEN+1, 74\$	1001	
06		69		69	E8	003D5	BLBS	(R9), 72\$	1003	
	63	A7		02	E1	003D8	BBC	#2, (R9), 73\$		
				02	90	003DC	72\$: MOVB	#2, 99(RAB)	1005	
	63	A7		04	11	003E0	BRB	74\$		
		54	63	01	90	003E2	73\$: MOVB	#1, 99(RAB)	1007	
	10	AE		A7	9E	003E6	74\$: MOVAB	99(RAB), R4	1008	
OB	2D	AE		64	90	003EA	MOVB	(R4), USR RFM		
		04		05	E0	003EE	BBS	#5, KEYWORDS_SEEN+1, 76\$	1014	
04		69		69	E8	003F3	BLBS	(R9), 75\$	1016	
	62	A7		02	E1	003F6	BBC	#2, (R9), 76\$		
09	2E	AE		02	88	003FA	75\$: BISB2	#2, 98(RAB)	1018	
04	48	A7		03	E0	003FE	76\$: BBS	#3, KEYWORDS_SEEN+2, 77\$	1024	
	FC	A7		04	E0	00403	BBS	#4, 72(RAB), 77\$	1026	
		05		20	88	00408	BISB2	#32, -4(RAB)	1028	
			2E	AE	E8	0040C	77\$: BLBS	KEYWORDS_SEEN+2, 78\$	1034	
			61	A7	94	00410	CLRB	97(RAB)	1036	
		OD		10	11	00413	BRB	80\$		
			61	69	E9	00415	78\$: BLBC	(R9), 80\$	1037	
				A7	95	00418	TSTB	97(RAB)		
				08	13	0041B	BEQL	80\$		
		7E	00G	7E	D4	0041D	79\$: CLRL	-(SP)	1039	
				8F	9A	0041F	MOVZBL	#PASSK_TEXREQSEQ, -(SP)		
		53	61	7F	11	00423	BRB	90\$		
14		AE		A7	9E	00425	80\$: MOVAB	97(RAB), R3	1040	
		14	2E	63	90	00429	MOVB	(R3), USR ORG		
				AE	E9	0042D	BLBC	KEYWORDS_SEEN+2, 82\$	1051	
		09	5B	12	12	00431	BNEQ	82\$	1052	
				A7	E8	00433	BLBS	91(RAB), 81\$	1054	



05	5B	0A	5A	A7	E9	00437	BLBC	90(RAB), 82\$	1055
	5B	A7		01	E1	0043B	BBC	#1, 91(RAB), 82\$	
0F	2F	AE	40	8F	88	00440	BISB2	#64, 91(RAB)	1057
06	FC	A7		03	E0	00445	BBS	#3, KEYWORDS_SEEN+3, 84\$	1059
	5B	A7		03	E1	0044A	BBC	#3, -4(RAB), -83\$	1061
				02	88	0044F	BISB2	#2, 91(RAB)	1063
	5B	A7		04	11	00453	BRB	84\$	
				20	88	00455	BISB2	#32, 91(RAB)	1065
21		58	FC	A7	9E	00459	MOVAB	-4(RAB), R8	1075
		68		04	E0	0045D	BBS	#4, (R8), 88\$	
0E		1E		68	E8	00461	BLBS	(R8), 88\$	1078
		68		01	E1	00464	BBC	#1, (R8), 85\$	1079
		10		63	91	00468	CMPB	(R3), #16	1080
				15	13	0046B	BEQL	88\$	
				63	95	0046D	TSTB	(R3)	1081
				05	12	0046F	BNEQ	85\$	
		01		64	91	00471	CMPB	(R4), #1	
				0C	13	00474	BEQL	88\$	
03		68		02	E0	00476	BBS	#2, (R8), 87\$	1082
				031E	31	0047A	BRW	146\$	
		20		63	91	0047D	CMPB	(R3), #32	1083
				F8	12	00480	BNEQ	86\$	
		5A	48	A7	9E	00482	MOVAB	72(RAB), R10	1091
1C		6A		04	E1	00486	BBC	#4, (R10), 91\$	
10		68		05	E0	0048A	BBS	#5, (R8), 89\$	1092
		0C	01	A8	E8	0048E	BLBS	1(R8), 89\$	
				68	95	00492	TSTB	(R8)	
				08	19	00494	BLSS	89\$	
04		68		04	E0	00496	BBS	#4, (R8), 89\$	1093
08		6A		19	E1	0049A	BBC	#25, (R10), 91\$	
				7E	D4	0049E	CLRL	-(SP)	1095
		7E	00G	8F	9A	004A0	MOVZBL	#PASSK_FILNAMREQ, -(SP)	
				18	11	004A4	BRB	93\$	
		18		69	E8	004A6	BLBS	(R9), 94\$	1102
14		68		04	E0	004A9	BBS	#4, (R8), 94\$	
		01		64	91	004AD	CMPB	(R4), #1	1103
				0F	12	004B0	BNEQ	94\$	
		65	08	AE	B1	004B2	CMPW	USR_USZ, (R5)	1104
				09	13	004B6	BEQL	94\$	
				7E	D4	004B8	CLRL	-(SP)	1106
		7E	00G	8F	9A	004BA	MOVZBL	#PASSK_RECLEINIC, -(SP)	
				02E0	31	004BE	BRW	147\$	
		50	08	A6	D0	004C1	MOVL	8(PFV), PFD	1116
				60	D5	004C5	TSTL	(PFD)	1118
				12	13	004C7	BEQL	95\$	
			F4	AD	9F	004C9	PUSHAB	XABKEY_SIZE	1121
			F8	AD	9F	004CC	PUSHAB	XABKEY_ADDR	
			BC	AD	9F	004CF	PUSHAB	XAB_SUM	
			00	B040	9F	004D2	PUSHAB	20(PFD)[PFD]	1122
0000V	CF			04	FB	004D6	CALLS	#4, FILL_KEY_XABS	1121
	6A		0420	8F	A8	004DB	BISW2	#1056, (R10)	1133
05	A7			06	88	004E0	BISB2	#6, 5(RAB)	1136
				63	95	004E4	TSTB	(R3)	1142
				0A	12	004E6	BNEQ	96\$	
		01		64	91	004E8	CMPB	(R4), #1	1143
				05	13	004EB	BEQL	96\$	
			2C	AE	95	004ED	TSTB	KEYWORDS_SEEN	1144

14	7A 2F	A7 AE	08	05 AE 02 56 57 58	18 B0 E1 DD DD DD	004F0 004F2 004F7 004FC 004FE 00500	96\$: 97\$:	BGEQ MOVW BBC PUSHL PUSHL PUSHL	97\$ USR USZ, 122(RAB) #2, -KEYWORDS_SEEN+3, 98\$ PFV RAB R11	1146 1158 1167
	00	51 B2 52	04	A2 03 50	DD FB DD	00502 00506 0050A		MOVL CALLS MOVL	4(USER_ACTION_BP), R1 #3, @0USER_ACTION_BP) R0, STATUS	1158
08		68	01	31 04 58	31 E1 DD	0050D 00510 00514	98\$:	BRW BBC PUSHL	118\$ #4, (R8), 99\$ R11	1181 1183
	00000000G	00		01 09 58	FB 11 DD	00516 0051D 0051F	99\$:	CALLS BRB PUSHL	#1, SYSS\$OPEN 100\$ R11	1185
	00000000G	00		01 50	FB E8	00521 00528	100\$:	CALLS BLBS	#1, SYSS\$CREATE \$\$STATUS, 101\$	1186
	0001825A	8F		50 DC	D1 13	0052B 00532		CMPL BEQL	\$\$STATUS, #98906 98\$	
		52 45		50 52	DD E8	00534 00537	101\$:	MOVL BLBS	\$\$STATUS, STATUS STATUS, 105\$	1187 1195
	0001829A	8F		52 36	D1 12	0053A 00541		CMPL BNEQ	STATUS, #98970 104\$	1197
04 2E		68 6A 2A	5A	04 19 A7	E0 E1 E9	00543 00547 0054B	102\$:	BBS BBC BLBC	#4, (R8), 102\$ #25, (R10), 104\$ 90(RAB), 104\$	1198 1199
	5A 5B 03	A7 A7 AA		02 20 01	90 8A 88	0054F 00553 00557		MOVW BICB2 BISB2	#2, 90(RAB) #32, 91(RAB) #1, 3(R10)	1207 1208 1209
	00000000G	00		58 01	DD FB	0055B 0055D	103\$:	PUSHL CALLS	R11 #1, SYSS\$OPEN	1214
	18	AE	18	50	DD	00564		MOVL	R0, \$\$STATUS	1215
	0001825A	8F		50 E6	D1 13	0056C 00573		MOVAB CMPL	\$\$STATUS, R0 R0, #98906	
		52 03	18	AE 52	DD E8	00575 00579	104\$:	BEQL MOVL	103\$ \$\$STATUS, STATUS STATUS, 105\$	1216 1225
		50	00C5	31	0057C		105\$:	BLBS BRW	119\$ 132(RAB), R0	1235
04	03	A8 03	0084	C7 0E 01	9E E1 88	0057F 00584 00588		MOVAB BBC BISB2	#14, (R0), 106\$ #1, 3(R8)	1237 1245
		60		69	E8	0058C	106\$:	BLBS	(R9), 108\$	
F9		60	0094	31	0058F		107\$:	BRW	116\$	
F0	03 62	A8 A7		02 02	E1 88	00592 00596	108\$:	BBC BISB2	#2, (R0), 107\$ #2, 3(R8)	1246 1249
7F 7B		EC 60 60	5A	01 A7 1B	E1 E9 E1	0059A 0059F 005A3		BBC BLBC BBC	#1, 98(RAB), 107\$ 90(RAB), 107\$ #27, (R0), 116\$	1257 1258 1259
		60		06 58	E0 DD	005A7 005AB	109\$:	PUSHL	R11	1262
	00000000G	00		01	FB	005AD		CALLS	#1, SYSS\$CLOSE	
	0001825A	8F		50	E8	005B4		BLBS	\$\$STATUS, 110\$	
		E7 A7 64	03	50 04 A8	D1 12 E8	005B7 005BE 005C0	110\$:	CMPL BNEQ BLBS	\$\$STATUS, #98906 110\$ 3(R8), 109\$	1263 1264
				04 03	90 90	005C4 005C8		MOVW MOVW	#4, 98(RAB) #3, (R4)	

PASSOPEN2  
1-015

OPEN procedure  
PASS\$OPEN - Open a file

M 13  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 43  
(10)

0083	C7	02	90	005CB	MOV	#2, 131(RAB)	1265	
03	AA	01	88	005D0	BISB2	#1, 3(R10)	1266	
00000000G	00	5B	DD	005D4	PUSHL	R11	1267	
0001825A	0D	01	FB	005D6	CALLS	#1, SYSS\$OPEN		
	8F	50	E8	005DD	BLBS	\$\$\$STATUS, 112\$		
	E7	50	D1	005E0	CMPL	\$\$\$STATUS, #98906		
	52	04	12	005E7	BNEQ	112\$		
	29	A8	E8	005E9	BLBS	3(R8), 111\$		
62	A7	50	DD	005ED	MOVL	\$\$\$STATUS, STATUS	1274	
	64	52	E8	005F0	BLBS	STATUS, 115\$	1277	
		02	90	005F3	MOV	#2, 98(RAB)	1278	
		02	90	005F7	MOV	#2, (R4)	1279	
		0083	C7	94	005FA	CLRB	131(RAB)	1281
			5B	DD	005FE	PUSHL	R11	
00000000G	00	01	FB	00600	CALLS	#1, SYSS\$OPEN		
0001825A	0D	50	E8	00607	BLBS	\$\$\$STATUS, 114\$		
	8F	50	D1	0060A	CMPL	\$\$\$STATUS, #98906		
	E7	04	12	00611	BNEQ	114\$		
	52	A8	E8	00613	BLBS	3(R8), 113\$		
		50	DD	00617	MOVL	\$\$\$STATUS, STATUS	1274	
		0A	11	0061A	BRB	116\$	1290	
01	A8	8F	88	0061C	BISB2	#64, 1(R8)	1291	
2C	A7	A7	9E	00621	MOVAB	-6(R7), 44(RAB)	1301	
	1B	52	E9	00626	BLBC	STATUS, 119\$	1303	
		57	DD	00629	PUSHL	RAB		
00000000G	00	01	FB	0062B	CALLS	#1, SYSS\$CONNECT		
	52	50	DD	00632	MOVL	R0, STATUS		
	75	52	E8	00635	BLBS	STATUS, 125\$	1304	
0001825A	8F	52	D1	00638	CMPL	STATUS, #98906		
	69	E8	13	0063F	BEQL	117\$		
00018292	8F	52	E8	00641	BLBS	STATUS, 125\$	1310	
		52	D1	00644	CMPL	STATUS, #98962	1315	
000184C4	8F	12	13	0064B	BEQL	120\$		
		52	D1	0064D	CMPL	STATUS, #99524		
0001C04A	8F	09	13	00654	BEQL	120\$		
		52	D1	00656	CMPL	STATUS, #114762		
	7E	06	12	0065D	BNEQ	121\$		
		8F	9A	0065F	MOVZBL	#PASSK_FILNOTFOU, -(SP)	1316	
000184CC	8F	40	11	00663	BRB	124\$		
		52	D1	00665	CMPL	STATUS, #99532	1318	
0001852C	8F	2D	13	0066C	BEQL	122\$		
		52	D1	0066E	CMPL	STATUS, #99628		
000185F4	8F	24	13	00675	BEQL	122\$		
		52	D1	00677	CMPL	STATUS, #99828		
000186D4	8F	1B	13	0067E	BEQL	122\$		
		52	D1	00680	CMPL	STATUS, #100052		
000186E4	8F	12	13	00687	BEQL	122\$		
		52	D1	00689	CMPL	STATUS, #100068		
000186FC	8F	09	13	00690	BEQL	122\$		
		52	D1	00692	CMPL	STATUS, #100092		
		06	12	00699	BNEQ	123\$		
	7E	8F	9A	0069B	MOVZBL	#PASSK_INVFILSYN, -(SP)	1319	
		04	11	0069F	BRB	124\$		
	7E	8F	9A	006A1	MOVZBL	#PASSK_ERRDUROPE, -(SP)	1322	
00000000G	00	01	FB	006A5	CALLS	#1, PASS\$SIGNAL		
		04		006AC	RET			
0D	6A	19	E1	006AD	BBC	#25, (R10), 126\$	1332	

00010619	8F	4C	A7	D1	006B1	CMPL	76(RAB), #67097	
			03	13	006B9	BEQL	126\$	
03	68		10	88	006BB	BISB2	#16, (R8)	1334
	68		04	E0	006BE	BBS	#4, (R8), 127\$	1340
		00C7	31	006C2	BRW	145\$		
	0E	2E	AE	E9	006C5	BLBC	KEYWORDS_SEEN+2, 128\$	1348
14	AE		63	91	006C9	CMPB	(R3), USR_ORG	1349
			08	13	006CD	BEQL	128\$	
			7E	D4	006CF	CLRL	-(SP)	1351
	7E	00G	8F	9A	006D1	MOVZBL	#PASSK_ORGSPEINC, -(SP)	
			26	11	006D5	BRB	131\$	
	07		69	E9	006D7	BLBC	(R9), 129\$	1353
			63	95	006DA	TSTB	(R3)	
			03	13	006DC	BEQL	129\$	
1A	2D	FD3C	31	006DE	BRW	79\$		
	AE		03	E1	006E1	BBC	#3, KEYWORDS_SEEN+1, 132\$	1361
	64	10	AE	91	006E6	CMPB	USR_RFM, (R4)	1364
			14	13	006EA	BEQL	132\$	
	02	10	AE	91	006EC	CMPB	USR_RFM, #2	1366
			05	12	006F0	BNEQ	130\$	
	03		64	91	006F2	CMPB	(R4), #3	1367
			09	13	006F5	BEQL	132\$	
			7E	D4	006F7	CLRL	-(SP)	1369
	7E	00G	8F	9A	006F9	MOVZBL	#PASSK_RECTYPINC, -(SP)	
11	0087	00A1	31	006FD	BRW	147\$		
0B	0084		04	E0	00700	BBS	#4, 135(RAB), 133\$	1380
			02	E0	00706	BBS	#2, 132(RAB), 133\$	1381
		7A	A7	B5	0070C	TSTW	122(RAB)	1383
			06	12	0070F	BNEQ	133\$	
	7A	0080	C7	B0	00711	MOVW	128(RAB), 122(RAB)	1385
	52	7A	A7	3C	00717	MOVZWL	122(RAB), R2	1387
			5B	D4	0071B	CLRL	R11	
			52	D5	0071D	TSTL	R2	
			38	13	0071F	BEQL	141\$	
			5B	D6	00721	INCL	R11	
	01		64	91	00723	CMPB	(R4), #1	1389
			18	12	00726	BNEQ	137\$	
	07		69	E9	00728	BLBC	(R9), 134\$	1392
		2C	AE	95	0072B	TSTB	KEYWORDS_SEEN	1395
			29	18	0072E	BGEQ	141\$	
			05	11	00730	BRB	135\$	1396
	52		65	B1	00732	CMPW	(R5), R2	1400
			06	12	00735	BNEQ	136\$	
	52	08	AE	B1	00737	CMPW	USR_USZ, R2	1401
			1C	13	0073B	BEQL	141\$	
		FD78	31	0073D	BRW	92\$		1403
	04		69	E8	00740	BLBS	(R9), 138\$	1407
07	69		02	E1	00743	BBC	#2, (R9), 139\$	
		2C	AE	95	00747	TSTB	KEYWORDS_SEEN	1410
			0D	18	0074A	BGEQ	141\$	
			05	11	0074C	BNEQ	140\$	1411
	52		65	B1	0074E	CMPW	(R5), R2	1415
			EA	1A	00751	BGTRU	136\$	
	52	08	AE	B1	00753	CMPW	USR_USZ, R2	1416
			E4	1A	00757	BGTRU	136\$	
	12		69	E9	00759	BLBC	(R9), 143\$	1425
	05		5B	E9	0075C	BLBC	R11, 142\$	1427



		65		52	B0	0075F	MOVW	R2, (R5)	1429	
				0A	11	00762	BRB	143\$		
		65	D2	AD	B1	00764	142\$:	CMPW	XAB FHC+10, (R5)	1432
				04	1B	00768		BLEQU	143\$	
		65	D2	AD	B0	0076A		MOVW	XAB FHC+10, (R5)	1434
		1B		68	E8	0076E	143\$:	BLBS	(R8), 145\$	1442
OE		68		01	E1	00771		BBC	#1, (R8), 144\$	1443
		10		63	91	00775		CMPB	(R3), #16	1444
				12	13	00778		BEQL	145\$	
				63	95	0077A		TSTB	(R3)	1445
				05	12	0077C		BNEQ	144\$	
		01		64	91	0077E		CMPB	(R4), #1	
				09	13	00781		BEQL	145\$	
14		68		02	E1	00783	144\$:	BBC	#2, (R8), 146\$	1446
		20		63	91	00787		CMPB	(R3), #32	1447
				0F	12	0078A		BNEQ	146\$	
19		68		01	E1	0078C	145\$:	BBC	#1, (R8), 148\$	1458
05	0087	C7		04	E1	00790		BBC	#4, 135(RAB), 146\$	1460
		0E	00CA	C7	E9	00796		BLBC	202(RAB), 148\$	1461
				7E	D4	0079B	146\$:	CLRL	-(SP)	1463
		7E	00G	8F	9A	0079D		MOVZBL	#PASSK ACCMETINC, -(SP)	
	0G000000G	00		02	FB	007A1	147\$:	CALLS	#2, PASS\$SIGNAL	
				04	007A8			RET		
		20		63	91	007A9	148\$:	CMPB	(R3), #32	1470
				0E	12	007AC		BNEQ	149\$	
			F4	AD	9F	007AE		PUSHAB	XABKEY_SIZE	1473
			F8	AD	9F	007B1		PUSHAB	XABKEY_ADDR	
			BC	AD	9F	007B4		PUSHAB	XAB_SUM	
	0000V	CF		03	FB	007B7		CALLS	#3, -CHECK_KEY_XABS	
	FFFFFFFF	8F	0C	AE	D1	007BC	149\$:	CMPL	FILE_TYPE, #T	1483
				1A	12	007C4		BNEQ	150\$	
		13	00000000'	EF	E8	007C6		BLBS	PASS\$GV INPUT OPENED, 150\$	
00	BE	01	00000000'	01	90	007CD		MOVW	#1, PASS\$GV_INPUT_OPENED	1486
		01		10	00	FO 0C7D4		INSV	#0, #16, #1, 20(SP)	1487
				A8	0C	88 007DA		BISB2	#12, 1(R8)	1489
					42	11 007DE		BRB	152\$	1483
		01	0C	AE	D1	007E0	150\$:	CMPL	FILE_TYPE, #1	1491
				2A	12	007E4		BNEQ	151\$	
		23	00000000'	EF	E8	007E6		BLBS	PASS\$GV OUTPUT OPENED, 151\$	
	00000000'	01		01	90	007ED		MOVW	#1, PASS\$GV_OUTPUT_OPENED	1494
				A8	30	88 007F4		BISB2	#48, 1(R8)	1496
00	BE	01		12	01	FO 007F8		INSV	#1, #18, #1, 20(SP)	1497
00	BE	01		10	01	FO 007FE		INSV	#1, #16, #1, 20(SP)	1498
00	BE	01		11	00	FO 00804		INSV	#0, #17, #1, 20(SP)	1499
		04		A7	02	88 0080A		BISB2	#2, 4(RAB)	1500
					12	11 0080E		BRB	152\$	1491
00	BE	01		12	00	FO 00810	151\$:	INSV	#0, #18, #1, 20(SP)	1507
00	BE	01		10	01	FO 00816		INSV	#1, #16, #1, 20(SP)	1508
00	BE	01		11	00	FO 0081C		INSV	#0, #17, #1, 20(SP)	1509
				52	A7	9E 00822	152\$:	MOVAB	36(RAB), R2	1519
		13		69	E9	00826		BLBC	(R9), 153\$	1516
		7E		65	3C	00829		MOVZWL	(R5), -(SP)	1519
	00000000G	00		01	FB	0082C		CALLS	#1, PASS\$GET_VM	
		62		50	D0	00833		MOVL	R0, (R2)	
		02		08	88	00836		BISB2	#8, 2(R8)	1520
				0D	11	0083A		BRB	155\$	1516
06		69		02	E1	0083C	153\$:	BBC	#2, (R9), 154\$	1522

62		66		02	C1	00840	ADDL3	#2, (PFV), (R2)	1524
		62		03	11	00844	BRB	155\$	1526
	E8	A7		66	D0	00846	154\$:	MOVL	(PFV), (R2)
	EC	A7		62	D0	00849	155\$:	MOVL	(R2), -24(RAB)
	F4	A7		62	D0	0084D		MOVL	(R2), -20(RAB)
FO	A7	EC		65	3C	00851		MOVZWL	(R5), -12(RAB)
			F4	A7	C1	00855		ADDL3	-12(RAB), -20(RAB), -16(RAB)
			0097	C7	9E	0085C		MOVAB	151(RAB), R2
08		52		04	E1	00861		BBC	#4, (R10), 156\$
		6A	009F	C7	94	00865		CLRB	159(RAB)
				62	94	00869		CLRB	(R2)
				1E	11	0086B		BRB	157\$
		7F		62	9A	0086D	156\$:	MOVZBL	(R2), -(SP)
00000000G		00		01	FB	00870		CALLS	#1, PAS\$GET_VM
04		BE		50	D0	00877		MOVL	R0, #4(SP)
		50		62	9A	0087B		MOVZBL	(R2), R0
		56	04	BE	D0	0087E		MOVL	#4(SP), R6
66	34	AE		50	28	00882		MOVCL3	R0, RESULT_NAME_STRING, (R6)
	02	A8		04	88	00887		BISB2	#4, 2(R8)
	E0	A7		01	CE	0088B	157\$:	MNEGL	#1, -32(RAB)
		4C		69	E9	0088F		BLBC	(R9), 158\$
	24	AE	01CE000D	8F	D0	00892		MOVL	#17694733, LOGNAM_DSC
	28	AE	F755	CF	9E	0089A		MOVAB	P.AAG, LOGNAM_DSC+4
	1C	AE	010E00FF	8F	D0	008A0		MOVL	#17694975, LIMIT_DSC
	20	AE	34	AE	9E	008A8		MOVAB	RESULT_NAME_STRING, LIMIT_DSC+4
				7E	7C	008AD		CLRL	-(SP)
				7E	D4	008AF		CLRL	-(SP)
			28	AE	9F	008B1		PUSHAB	LIMIT_DSC
			2C	AE	9F	008B4		PUSHAB	LIMIT_DSC
			38	AE	9F	008B7		PUSHAB	LOGNAM_DSC
00000000G		00		06	FB	008BA		CALLS	#6, SYS\$STRNLOG
		01		50	D1	008C1		CMPL	R0, #1
				18	12	008C4		BNEQ	158\$
				01	DD	008C6		PUSHL	#1
				04	DD	008C8		PUSHL	#4
			E0	A7	9F	008CA		PUSHAB	-32(RAB)
			28	AE	9F	008CD		PUSHAB	LIMIT_DSC
00000000G		00		04	FB	008D0		CALLS	#4, OT\$SCVT_TI_L
		04		50	E8	008D7		BLBS	R0, 158\$
	E0	A7		01	CE	008DA		MNEGL	#1, -32(RAB)
			68	A7	D4	008DE	158\$:	CLRL	104(RAB)
			F8	AD	D5	008E1		TSTL	XABKEY_ADDR
				0D	13	008E4		BEQL	159\$
			F8	AD	9F	008E6		PUSHAB	XABKEY_ADDR
			F4	AD	DD	008E9		PUSHL	XABKEY_SIZE
00	BE			02	FB	008EC		CALLS	#2, PAS\$FREE_VM
		00		01	F0	008F3	159\$:	INSV	#1, #29, #1, #0(SP)
		1D		57	DD	008F9		PUSHL	RAB
		00		01	FB	008FB		CALLS	#1, PAS\$ADD_FILE
				04	00902			RET	
				0000	00903	160\$:		.WORD	Save nothing
		50	08	AC	D0	00905		MOVL	8(AP), R0
		50	04	A0	D0	00909		MOVL	4(R0), R0
			F4	A0	9F	0090D		PUSHAB	XABKEY_SIZE
			F8	A0	9F	00910		PUSHAB	XABKEY_ADDR
			FC	A0	9F	00913		PUSHAB	PFV_ADDR
				03	DD	00916		PUSHL	#3

PASSOPEN2  
1-015

OPEN procedure  
PASS\$OPEN - Open a file

D 14  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 47  
(10)

			5E	DD	00918	PUSHL	SP	
			AC	7D	0091A	MOVQ	4(AP), -(SP)	
			03	FB	0091E	CALLS	#3, OPEN_HANDLER	
			04	00923	RET			

: Routine Size: 2340 bytes, Routine Base: \_PASS\$CODE + 00E5

: 1585 1639 1  
: 1586 1640 1 !<BLF/PAGE>

```
1588 1641 1 %SBTTL 'PAS$OPEN IMPLICIT - Open implicitly opened files'
1589 1642 1 GLOBAL ROUTINE PAS$OPEN_IMPLICIT (
1590 1643 1     PFV: REF $PAS$PFV FILE VARIABLE,      ! File variable
1591 1644 1     IN_FCB: REF $PAS$FCB CONTROL_BLOCK;    ! Control block
1592 1645 1     FCB: REF $PAS$FCB CONTROL_BLOCK      ! Output FCB
1593 1646 1 ): JSB_OPEN_IMPLICIT NOVALUE =
1594 1647 1
1595 1648 1 ++
1596 1649 1 FUNCTIONAL DESCRIPTION:
1597 1650 1
1598 1651 1     This procedure is called from procedures which do not
1599 1652 1     accept textfiles, but have been called with a file that
1600 1653 1     has its PFV$V_VALID bit clear. So that a proper error
1601 1654 1     message can be given, this procedure opens the file INPUT
1602 1655 1     or OUTPUT, if that is the current file and if that file
1603 1656 1     has not already been opened.
1604 1657 1
1605 1658 1 CALLING SEQUENCE:
1606 1659 1
1607 1660 1     JSB_OPEN_IMPLICIT PAS$OPEN_IMPLICIT (PFV.mr.r, IN_FCB.mr.r;
1608 1661 1     FCB.mr.r)
1609 1662 1
1610 1663 1 FORMAL PARAMETERS:
1611 1664 1
1612 1665 1     PFV          - The Pascal File Variable (PFV) for the file.
1613 1666 1
1614 1667 1     IN_FCB       - The File Control Block (FCB) for the file.
1615 1668 1
1616 1669 1     FCB          - The result FCB for the file.
1617 1670 1
1618 1671 1 IMPLICIT INPUTS:
1619 1672 1
1620 1673 1     It is assumed that the caller has verified the PFV and has locked
1621 1674 1     it.
1622 1675 1
1623 1676 1 IMPLICIT OUTPUTS:
1624 1677 1
1625 1678 1     NONE
1626 1679 1
1627 1680 1 COMPLETION STATUS:
1628 1681 1
1629 1682 1     NONE
1630 1683 1
1631 1684 1 SIDE EFFECTS:
1632 1685 1
1633 1686 1     NONE
1634 1687 1
1635 1688 1 SIGNALLED ERRORS:
1636 1689 1
1637 1690 1
1638 1691 1 --
1639 1692 1
1640 1693 2 BEGIN
1641 1694 2
1642 1695 2     FCB = IN_FCB [FCB$R_FCB];      ! Set output FCB
1643 1696 2
1644 1697 2 !+
```



```
1645 1698 2 ! See if the file is open. If not, but is INPUT or OUTPUT, open it.
1646 1699 2 ! Otherwise, return.
1647 1700 2 !-
1648 1701 2
1649 1702 2 IF NOT .PFV [PFV$V_OPEN] ! Not open
1650 1703 2 THEN
1651 1704 2 BEGIN
1652 1705 2 IF (PFV [PFV$R_PFV] EQLA PASS$FV_INPUT) AND NOT .PASS$GV_INPUT_OPENED
1653 1706 2 THEN
1654 1707 2 PASS$OPEN (PFV [PFV$R_PFV], PASS$K_HISTORY_READONLY; FCB)
1655 1708 2 ELSE IF (PFV [PFV$R_PFV] EQLA PASS$FV_OUTPUT) AND NOT .PASS$GV_OUTPUT_OPENED
1656 1709 2 THEN
1657 1710 2 PASS$OPEN (PFV [PFV$R_PFV], PASS$K_HISTORY_NEW; FCB)
1658 1711 2 ELSE
1659 1712 2 BEGIN
1660 1713 2 !-
1661 1714 2 ! If PFD or buffer address is self-relative, resolve it.
1662 1715 2 !-
1663 1716 2
1664 1717 2 IF .PFV [PFV$V_RELPFD]
1665 1718 2 THEN
1666 1719 2 BEGIN
1667 1720 2 PFV [PFV$A_PFD] = .PFV [PFV$A_PFD] + PFV [PFV$R_PFV];
1668 1721 2 PFV [PFV$V_RELPFD] = 0;
1669 1722 2 END;
1670 1723 2
1671 1724 2 IF .PFV [PFV$V_RELBUF]
1672 1725 2 THEN
1673 1726 2 BEGIN
1674 1727 2 PFV [PFV$A_BUFFER] = .PFV [PFV$A_BUFFER] + PFV [PFV$R_PFV];
1675 1728 2 PFV [PFV$V_RELBUF] = 0;
1676 1729 2 END;
1677 1730 2
1678 1731 2 END;
1679 1732 2 END;
1680 1733 2 RETURN;
1681 1734 2
1682 1735 2
1683 1736 1 END;

! End of routine PAS$OPEN_IMPLICIT
```

52	04	A6	9E	00000	PAS\$OPEN_IMPLICIT::		
62		1D	E0	00004	MOVAB	4(PFV), R2	1702
50	00000000G	00	9E	00008	BBS	#29, (R2), 5\$	
50		56	D1	0000F	MOVAB	PAS\$FV_INPUT, R0	1705
		0B	12	00012	CMPL	PFV, R0	
04	00000000'	EF	E8	00014	BNEQ	1\$	
		06	DD	0001B	BLBS	PAS\$GV_INPUT_OPENED, 1\$	
		15	11	0001D	PUSHL	#6	1707
50	00000000G	00	9E	0001F	BRB	2\$	
50		56	D1	00026	MOVAB	PAS\$FV_OUTPUT, R0	1708
		11	12	00029	CMPL	PFV, R0	
0A	00000000'	EF	E8	0002B	BNEQ	3\$	
					BLBS	PAS\$GV_OUTPUT_OPENED, 3\$	

```
; Routine Size: 84 bytes,   Routine Base: _PASSCODE + 0A09
```

```
1685 1737 1 XSBTTL 'FILL_KEY_XABS - Fill in key XABs for indexed file'
1686 1738 1 ROUTINE FILL_KEY_XABS (
1687 1739 1     PFV: REF $PASSPFV_FILE_VARIABLE,      | File variable
1688 1740 1     FCB: REF $PASSFCB_CONTROL_BLOCK,      | File control block
1689 1741 1     KDB: REF $PASSKDB_KEY_DESCRIPTOR,    | Key descriptor
1690 1742 1     XAB_SUM: REF $XABSUM_DECL,          | Summary XAB
1691 1743 1     XABKEY_ADDR: REF VECTOR [, LONG],    | Address of KEY XABs
1692 1744 1     XABKEY_SIZE: REF VECTOR [, LONG],    | Length of KEY XABs
1693 1745 1 ): CALL_FILL_KEY_XABS NOVALUE =
1694 1746 1
1695 1747 1 ++
1696 1748 1 FUNCTIONAL DESCRIPTION:
1697 1749 1
1698 1750 1     This procedure is called by PASS$OPEN to allocate and fill in
1699 1751 1     the KEY XABs for an indexed file.
1700 1752 1
1701 1753 1 CALLING SEQUENCE:
1702 1754 1
1703 1755 1     CALL_FILL_KEY_XABS FILL_KEY_XABS (PFV.rr.r, FCB.mr.r,
1704 1756 1     KDB.rr.r, XAB_SUM.wr.r, XABKEY_ADDR.wa.r, XABKEY_SIZE.wl.r)
1705 1757 1
1706 1758 1 FORMAL PARAMETERS:
1707 1759 1
1708 1760 1     PFV          - Pascal File Variable
1709 1761 1
1710 1762 1     FCB          - File control Block
1711 1763 1
1712 1764 1     KDB          - Key descriptor block. The structure is:
1713 1765 1
1714 1766 1
1715 1767 1     +-----+-----+-----+-----+
1716 1768 1     |          MBZ          | Count | <-- KDB
1717 1769 1     +-----+-----+-----+-----+
1718 1770 1     | MBZ | Size | Dtype | Key # |
1719 1771 1     +-----+-----+-----+-----+
1720 1772 1     |          Byte offset into record          |
1721 1773 1     +-----+-----+-----+-----+
1722 1774 1     | MBZ | Size | Dtype | Key # |
1723 1775 1     +-----+-----+-----+-----+
1724 1776 1     |          Byte offset into record          |
1725 1777 1     +-----+-----+-----+-----+
1726 1778 1     |          ...          |
1727 1779 1
1728 1780 1     XAB_SUM      - The summary XAB allocated by our caller. The
1729 1781 1     key XABs will be chained to it.
1730 1782 1
1731 1783 1     XABKEY_ADDR  - Place to store address of allocated KEY XABs
1732 1784 1     so that they may be deallocated in case of an
1733 1785 1     error.
1734 1786 1
1735 1787 1     XABKEY_SIZE  - Place to store length of allocated KEY XABs.
1736 1788 1
1737 1789 1 IMPLICIT INPUTS:
1738 1790 1
1739 1791 1     NONE
1740 1792 1
1741 1793 1 IMPLICIT OUTPUTS:
```

```
1742 1794 1 |      NONE
1743 1795 1 |
1744 1796 1 | COMPLETION STATUS:
1745 1797 1 |
1746 1798 1 |      NONE
1747 1799 1 |
1748 1800 1 | SIDE EFFECTS:
1749 1801 1 |
1750 1802 1 |      NONE
1751 1803 1 |
1752 1804 1 | SIGNALLED ERRORS:
1753 1805 1 |
1754 1806 1 |
1755 1807 1 | --
1756 1808 1 |
1757 1809 2 | BEGIN
1758 1810 2 |
1759 1811 2 | LOCAL
1760 1812 2 |     NEXT_KEY: REF $PASSKDB_KEY_DESCRIPTOR, | Descriptor of next key
1761 1813 2 |     XAB_KEY: REF $XABKEY_DECL,             | Key XAB
1762 1814 2 |     LAST_XAB: REF $XABKEY_DECL,           | Previous key XAB
1763 1815 2 |     LAST_KEY: SIGNED;                     | Last key number
1764 1816 2 |
1765 1817 2 | NEXT_KEY = .KDB;                          | Get KDB address
1766 1818 2 |
1767 1819 2 | !+
1768 1820 2 | ! Allocate all the KEY XABs we need to describe the
1769 1821 2 | ! keys.
1770 1822 2 | !-
1771 1823 2 |
1772 1824 2 | FCB [FCB$L_NKEYS] = .NEXT_KEY [KDB$B_COUNT];
1773 1825 2 | XAB_SUM [XAB$B_NOK] = .NEXT_KEY [KDB$B_COUNT]; | RMS doesn't set on create
1774 1826 2 | XABKEY_SIZE [0] = .FCB [FCB$L_NKEYS] * XAB$C_KEYLEN;
1775 1827 2 | IF .XABKEY_SIZE [0] NEQ 0
1776 1828 2 | THEN
1777 1829 2 |     BEGIN
1778 1830 2 |         XAB_KEY = PASS$GET_VM (PFV [PFV$R_PFV], .XABKEY_SIZE [0]);
1779 1831 2 |         XABKEY_ADDR [0] = .XAB_KEY;
1780 1832 2 |     END;
1781 1833 2 | LAST_XAB = XAB_SUM [0,0,0,0];
1782 1834 2 |
1783 1835 2 | NEXT_KEY = NEXT_KEY [4,0,0,0]; | Advance pointer over count longword.
1784 1836 2 | LAST_KEY = -1; | Initialize for comparison
1785 1837 2 |
1786 1838 2 | !+
1787 1839 2 | ! For each key, fill in the KEY XAB.
1788 1840 2 | !-
1789 1841 2 |
1790 1842 2 | INCR KEY_NUM FROM 0 TO .FCB [FCB$L_NKEYS] - 1 DO
1791 1843 2 |     BEGIN
1792 1844 2 |
1793 1845 2 |         XAB_KEY [XAB$B_COD] = XAB$C_KEY;
1794 1846 2 |         XAB_KEY [XAB$B_BLN] = XAB$C_KEYLEN;
1795 1847 2 |         XAB_KEY [XAB$B_REF] = .NEXT_KEY [KDB$B_KEY_NUMBER];
1796 1848 2 |
1797 1849 2 |
1798 1850 2 | !+
```



```
1799 1851 3      ! Ensure that the keys are in ascending order.
1800 1852 3      !-
1801 1853 3      IF .XAB_KEY [XAB$B_REF] LEQ .LAST_KEY
1802 1854 3      THEN
1803 1855 3          $PASSIO_ERROR (PASS_INVARGPAS,0);
1804 1856 3
1805 1857 3      !+
1806 1858 3      ! If this is possibly a $CREATE, make sure that the keys
1807 1859 3      ! are in order and dense.
1808 1860 3      !-
1809 1861 3      IF NOT .FCB [FCB$V_OLD_FILE]
1810 1862 3      THEN
1811 1863 3          IF .XAB_KEY [XAB$B_REF] NEQ .KEY_NUM
1812 1864 3          THEN
1813 1865 3              $PASSIO_ERROR (PASS_INVKEYDEF,0);
1814 1866 3
1815 1867 3      IF .XAB_KEY [XAB$B_REF] NEQ 0      ! Not primary key?
1816 1868 3      THEN
1817 1869 3          XAB_KEY [XAB$B_FLG] = XAB$M_CHG+XAB$M_DUP;      ! Allow changes and duplicates
1818 1870 3
1819 1871 3      IF .NEXT_KEY [KDB$B_DTYPE] EQL DSC$K_DTYPE_T      ! String?
1820 1872 3      THEN
1821 1873 3          BEGIN
1822 1874 3              XAB_KEY [XAB$B_DTP] = XAB$C_STG;
1823 1875 3              XAB_KEY [XAB$B_SIZE] = .NEXT_KEY [KDB$B_SIZE];
1824 1876 3          END
1825 1877 3      ELSE IF .NEXT_KEY [KDB$B_DTYPE] EQL DSC$K_DTYPE_BU      ! Enumerated?
1826 1878 3      THEN
1827 1879 3          BEGIN
1828 1880 3              !+
1829 1881 3              ! Since RMS doesn't have an unsigned byte type,
1830 1882 3              ! call it a 1-byte string, which is equivalent.
1831 1883 3              !-
1832 1884 3              XAB_KEY [XAB$B_DTP] = XAB$C_STG;
1833 1885 3              XAB_KEY [XAB$B_SIZE] = 1;
1834 1886 3          END
1835 1887 3      ELSE CASE .NEXT_KEY [KDB$B_DTYPE] FROM DSC$K_DTYPE_WU TO DSC$K_DTYPE_L OF
1836 1888 3      SET
1837 1889 3      [DSC$K_DTYPE_WU]:
1838 1890 3          BEGIN
1839 1891 3              XAB_KEY [XAB$B_DTP] = XAB$C_BN2;
1840 1892 3              XAB_KEY [XAB$B_SIZE] = 2;
1841 1893 3          END;
1842 1894 3      [DSC$K_DTYPE_LU]:
1843 1895 3          BEGIN
1844 1896 3              XAB_KEY [XAB$B_DTP] = XAB$C_BN4;
1845 1897 3              XAB_KEY [XAB$B_SIZE] = 4;
1846 1898 3          END;
1847 1899 3      [DSC$K_DTYPE_W]:
1848 1900 3          BEGIN
1849 1901 3              XAB_KEY [XAB$B_DTP] = XAB$C_IN2;
1850 1902 3              XAB_KEY [XAB$B_SIZE] = 2;
1851 1903 3          END;
1852 1904 3      [DSC$K_DTYPE_L]:
1853 1905 3          BEGIN
1854 1906 3              XAB_KEY [XAB$B_DTP] = XAB$C_IN4;
1855 1907 3              XAB_KEY [XAB$B_SIZE] = 4;
```

```
1856 1908 3
1857 1909
1858 1910
1859 1911
1860 1912
1861 1913
1862 1914
1863 1915
1864 1916
1865 1917
1866 1918
1867 1919
1868 1920
1869 1921
1870 1922
1871 1923
1872 1924
1873 1925
1874 1926
1875 1927
1876 1928
1877 1929
1878 1930
1879 1931
1880 1932
1881 1933
```

```
END;
[INRANGE, OTRANGE]:
$PASSIO_ERROR (PASS_INVARGPAS,0);
TES;

!+
! Since if the file is created, RMS doesn't fill in TKS,
! do it here.
-
XAB_KEY [XAB$B_TKS] = .XAB_KEY [XAB$B_SIZE];
IF .NEXT_KEY [KDB$L_OFFSET] GTRU 65535
THEN
    $PASSIO_ERROR (PASS_INVKEYDEF,0);
XAB_KEY [XAB$W_POS0] = .NEXT_KEY [KDB$L_OFFSET];

LAST_XAB [XAB$L_NXT] = .XAB_KEY;
LAST_XAB = .XAB_KEY;
LAST_KEY = .XAB_KEY [XAB$B_REF];
XAB_KEY = .XAB_KEY + XAB$C_KEYLEN;
NEXT_KEY = NEXT_KEY [8,0,0,0]; ! Advance KDB
END;

RETURN;

END;
```

! End of routine FILL\_KEY\_XABS

.EXTRN PASSK\_INVKEYDEF

```
013C 00000 FILL_KEY_XABS:
WORD Save R2,R3,R4,R5,R8
DO 54 04 AC D0 00002 MOVL KDB, NEXT_KEY
A7 64 9A 00006 MOVZBL (NEXT_KEY), -48(FCB)
53 08 AC D0 0000A MOVL XAB$OM, R3
A3 64 90 0000E MOVBL (NEXT_KEY), 9(R3)
DO 10 BC D0 A7 0000004C 8F C5 00012 MULL3 #76, -48(FCB), @XABKEY_SIZE
11 13 0001C BEQL 1$
BC DD 0001E PUSHL @XABKEY_SIZE
00 01 FB 00021 CALLS #1, PASS$GET_VM
52 50 D0 00028 MOVL R0, XAB_KEY
OC BC 52 D0 0002B MOVL XAB_KEY, @XABKEY_ADDR
54 04 C0 0002F 1$: ADDL2 #4, NEXT_KEY
58 01 CE 00032 MNEGL #1, LAST_KEY
55 01 CE 00035 MNEGL #1, KEY_NUM
00B3 31 00038 BRW 20$
62 4C15 8F B0 0003B 2$: MOVW #19477, (XAB_KEY)
A2 64 90 00040 MOVBL (NEXT_KEY), 23(XAB_KEY)
17 08 00 ED 00044 CMPZV #0, #8, 23(XAB_KEY), LAST_KEY
58 17 A2 08 08 14 0004A BGTR 4$
7E D4 0004C 3$: CLRL -(SP)
7E 00G 8F 9A 0004E MOVZBL #PASSK_INVARGPAS, -(SP)
04 E0 00054 4$: BRB 18$
55 17 A2 FC A7 08 00 ED 00059 BBS #4, -4(FCB), 5$
08 00 ED 00059 CMPZV #0, #8, 23(XAB_KEY), KEY_NUM
68 12 0005F BNEQ 17$
```

PASSOPEN2  
1-015

OPEN procedure

FILL\_KEY\_XABS - Fill in key XABs for indexed fi

L 14

16-Sep-1984 01:46:15

14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 55  
(12)

FFBE

05  
FFBE

03  
0014  
0024

0000FFFF

00000000G

01

17 A2 95 00061 5\$:  
04 13 00064  
12 A2 03 90 00066  
0E 01 A4 91 0006A 6\$:  
0A 12 0006E  
13 A2 94 00070  
2E A2 02 A4 90 00073  
02 40 11 00078  
01 A4 91 0007A 7\$:  
09 12 0007E  
13 A2 94 00080  
2E A2 01 90 00083  
31 11 00087  
01 A4 8F 00089 8\$:  
000E 0008E 9\$:  
001A 00096  
  
B0 11 0009A  
02 90 0009C 10\$:  
0A 11 000A0  
13 A2 04 90 000A2 11\$:  
0E 11 000A6  
13 A2 01 90 000A8 12\$:  
2E A2 02 90 000AC 13\$:  
08 11 000B0  
13 A2 03 90 000B2 14\$:  
2E A2 04 90 000B6 15\$:  
16 A2 2E A2 90 000BA 16\$:  
0000FFFF 8F 04 A4 D1 000BF  
0E 1B 000C7  
7E D4 000C9 17\$:  
00G 8F 9A 000CB  
02 FB 000CF 18\$:  
04 04 000D6  
1E A2 04 A4 B0 000D7 19\$:  
04 A3 52 D0 000DC  
53 82 7E 000E0  
58 0F A2 9A 000E3  
52 44 A2 9E 000E7  
54 08 C0 000EB  
55 D0 A7 F2 000EE 20\$:  
04 000F3  
FF44 31 000F4 21\$:

TSTB 23(XAB\_KEY)  
BEQL 6\$  
MOVB #3, 18(XAB\_KEY)  
CMPB 1(NEXT\_KEY), #14  
BNEQ 7\$  
CLRB 19(XAB\_KEY)  
MOVB 2(NEXT\_KEY), 46(XAB\_KEY)  
BRB 16\$  
CMPB 1(NEXT\_KEY), #2  
BNEQ 8\$  
CLRB 19(XAB\_KEY)  
MOVB #1, 46(XAB\_KEY)  
BRB 16\$  
CASEB 1(NEXT\_KEY), #3, #5  
.WORD 10\$-9\$,-  
11\$-9\$,-  
3\$-9\$,-  
3\$-9\$,-  
12\$-9\$,-  
14\$-9\$  
3\$  
BRB #2, 19(XAB\_KEY)  
13\$  
MOVB #4, 19(XAB\_KEY)  
BRB 15\$  
MOVB #1, 19(XAB\_KEY)  
MOVB #2, 46(XAB\_KEY)  
BRB 16\$  
MOVB #3, 19(XAB\_KEY)  
MOVB #4, 46(XAB\_KEY)  
MOVB 46(XAB\_KEY), 22(XAB\_KEY)  
CML 4(NEXT\_KEY), #65535  
BLEQU 19\$  
CLRL -(SP)  
MOVZBL #PASSK\_INVKEYDEF, -(SP)  
CALLS #2, PASS\$SIGNAL  
RET  
MOVW 4(NEXT\_KEY), 30(XAB\_KEY)  
MOVL XAB\_KEY, 4(LAST\_XAB)  
MOVAQ (XAB\_KEY)+, LAST\_XAB  
MOVZBL 15(XAB\_KEY), LAST\_KEY  
MOVAB 68(R2), XAB\_KEY  
ADDL2 #8, NEXT\_KEY  
AOBLSS -48(FCB), KEY\_NUM, 21\$  
RET  
BRW 2\$

1867  
1869  
1871  
1874  
1875  
1871  
1877  
1884  
1885  
1877  
1887  
  
1910  
1891  
1892  
1896  
1897  
1901  
1902  
1887  
1906  
1907  
1917  
1919  
1921  
  
1922  
1924  
1925  
1926  
1927  
1928  
1842  
1933  
1842

; Routine Size: 247 bytes, Routine Base: \_PASSCODE + 0A5D

```
1883 1934 1 %SBTTL 'CHECK_KEY_XABS - Check key XABs for indexed file'
1884 1935 1 ROUTINE CHECK_KEY_XABS (
1885 1936 1     PFV: REF $PASS$PFV_FILE_VARIABLE,      ! Pascal File Variable
1886 1937 1     FCB: REF $PASS$FCB_CONTROL_BLOCK,    ! File Control Block
1887 1938 1     XAB_SUM: REF $XABSUM_DECL,          ! Summary XAB
1888 1939 1     XABKEY_ADDR: REF VECTOR [, LONG],    ! Address of KEY XABs
1889 1940 1     XABKEY_SIZE: REF VECTOR [, LONG],    ! Size of KEY XABs
1890 1941 1 ): CALL_CHECK_KEY_XABS NOVALUE =
1891 1942 1
1892 1943 1 ++
1893 1944 1 FUNCTIONAL DESCRIPTION:
1894 1945 1
1895 1946 1     This procedure is called by PASS$OPEN to compare the keys that
1896 1947 1     are declared in the KDB (if any) against those actually defined
1897 1948 1     for the indexed file.
1898 1949 1
1899 1950 1 CALLING SEQUENCE:
1900 1951 1
1901 1952 1     CALL_CHECK_KEY_XABS (PFV.r.r.r, FCB.mr.r, XAB_SUM.r.r.r,
1902 1953 1                          XABKEY_ADDR.ma.r, XABKEY_SIZE.ml.r)
1903 1954 1
1904 1955 1 FORMAL PARAMETERS:
1905 1956 1
1906 1957 1     PFV          - The Pascal File Variable (PFV) for the file.
1907 1958 1
1908 1959 1     FCB          - The result FCB for the file.
1909 1960 1
1910 1961 1     XAB_SUM      - The summary XAB. If new KEY XABs are allocated
1911 1962 1                   by this procedure, they are linked to XAB_SUM.
1912 1963 1
1913 1964 1     XABKEY_ADDR  - The address of the previously allocated KEY XABs.
1914 1965 1
1915 1966 1     XABKEY_SIZE  - The size of the previously allocated KEY XABs.
1916 1967 1
1917 1968 1 IMPLICIT INPUTS:
1918 1969 1
1919 1970 1     NONE
1920 1971 1
1921 1972 1 IMPLICIT OUTPUTS:
1922 1973 1
1923 1974 1     NONE
1924 1975 1
1925 1976 1 COMPLETION STATUS:
1926 1977 1
1927 1978 1     NONE
1928 1979 1
1929 1980 1 SIDE EFFECTS:
1930 1981 1
1931 1982 1     NONE
1932 1983 1
1933 1984 1 SIGNALLED ERRORS:
1934 1985 1
1935 1986 1
1936 1987 1 --
1937 1988 1
1938 1989 2 BEGIN
1939 1990 2
```



```
1940 1991 2 LOCAL
1941 1992 PFD: REF $PASSPFD_FILE_DESCRIPTOR, ! File descriptor
1942 1993 KDB: REF $PASSKDB_KEY_DESCRIPTOR, ! Key descriptor block
1943 1994 XAB_KEY: REF $XABKEY_DECL, ! Pointer to KEY XAB
1944 1995 KDB_NKEYS, ! Number of keys in KDB
1945 1996 KEYTYPES: REF VECTOR [, WORD]; ! Vector of allowable key datatypes and lengths
1946 1997
1947 1998
1948 1999 BIND
1949 2000 FAB = FCB: REF $PASSFAB_FCB_STRUCT;
1950 2001 FCB [FCB$V_INDEXED] = 1; ! Indicate indexed organization
1951 2002
1952 2003 !+
1953 2004 ! Get number of keys defined for file.
1954 2005 !-
1955 2006
1956 2007 FCB [FCB$L_NKEYS] = .XAB_SUM [XAB$B_NOK];
1957 2008
1958 2009 !+
1959 2010 ! Get number of user-defined keys from KDB (if any).
1960 2011 !-
1961 2012
1962 2013 PFD = .PFV [PFV$A_PFD];
1963 2014 IF .PFD [PFD$A_KDB] NEQA 0 ! Is there a KDB?
1964 2015 THEN
1965 2016 BEGIN
1966 2017 KDB = .PFD [PFD$A_KDB] + PFD [PFD$R_PFD]; ! Get address of KDB
1967 2018 KDB_NKEYS = .KDB [KDB$B_COUNT]; ! Get count
1968 2019 KDB = KDB [4,0,0,0]; ! Advance to first key description
1969 2020 END
1970 2021 ELSE
1971 2022 KDB_NKEYS = 0;
1972 2023
1973 2024 !+
1974 2025 ! If the number of keys specified in the KDB doesn't match the
1975 2026 ! actual number in the file, we have to throw away the KEY XABs
1976 2027 ! we allocated, allocate the correct number, and do a $DISPLAY
1977 2028 ! to fill them in.
1978 2029 !-
1979 2030
1980 2031 IF .KDB_NKEYS NEQ .FCB [FCB$L_NKEYS]
1981 2032 THEN
1982 2033 BEGIN
1983 2034 LOCAL
1984 2035 LAST_XAB: REF BLOCK [, BYTE]; ! Previous XAB
1985 2036
1986 2037 !+
1987 2038 ! Deallocate the old KEY XABs.
1988 2039 !-
1989 2040
1990 2041 IF .XABKEY_ADDR [0] NEQA 0
1991 2042 THEN
1992 2043 PASS$FREE_VM (.XABKEY_SIZE [0], XABKEY_ADDR [0]);
1993 2044
1994 2045 !+
1995 2046 ! Allocate enough new KEY XABs.
1996 2047 !-
```

```
1997 2048 3
1998 2049
1999 2050 XABKEY_SIZE [0] = XAB$C_KEYLEN * .FCB [FCB$N_KEYS];
2000 2051 XAB_KEY = PASS$GET_VM (PFV [PFV$R_PFV], .XABKEY_SIZE [0]);
2001 2052 XABKEY_ADDR [0] = .XAB_KEY;
2002 2053
2003 2054 !+
2004 2055 ! Fill in each KEY XAB and link it into the chain.
2005 2056 !-
2006 2057 LAST_XAB = .XAB_SUM; ! Link KEY XABs after SUM XAB
2007 2058
2008 2059 INCR KEY_NUM FROM 0 TO .FCB [FCB$N_KEYS] - 1 DO
2009 2060 BEGIN
2010 2061 XAB_KEY [XAB$B_COD] = XAB$C_KEY;
2011 2062 XAB_KEY [XAB$B_BLN] = XAB$C_KEYLEN;
2012 2063 XAB_KEY [XAB$B_REF] = .KEY_NUM;
2013 2064 LAST_XAB [XAB$C_NXT] = .XAB_KEY; ! Link in XAB
2014 2065 LAST_XAB = .XAB_KEY;
2015 2066 XAB_KEY = .XAB_KEY + XAB$C_KEYLEN;
2016 2067 END;
2017 2068
2018 2069 !+
2019 2070 ! Make sure that XAB_SUM is linked to the FAB. A USER_ACTION
2020 2071 ! routine may have unlinked it.
2021 2072 !-
2022 2073
2023 2074 FAB [FAB$L_XAB] = .XAB_SUM;
2024 2075
2025 2076 ! Ask RMS to fill in our new XABs.
2026 2077 !-
2027 2078
2028 2079 IF NOT $PASSRMS_OP ($DISPLAY (FAB = FAB [0,0,0,0]))
2029 2080 THEN
2030 2081 $PASSIO_ERROR (PASS_ERRDUROPE);
2031 2082
2032 2083 END;
2033 2084
2034 2085 !+
2035 2086 ! Allocate a vector of longwords, one for each key, which will
2036 2087 ! contain a bit mask that indicates what key expression datatypes
2037 2088 ! may be used to reference that key, and the key size. The mask
2038 2089 ! occupies the first word and the length, the second word.
2039 2090 !-
2040 2091
2041 2092 KEYTYPES = PASS$GET_VM (PFV [PFV$R_PFV], .FCB [FCB$N_KEYS] * 4);
2042 2093 FCB [FCB$A_KEY_TYPES] = .KEYTYPES;
2043 2094
2044 2095 !+
2045 2096 ! For each key in the file, if the KDB describes the key, make sure
2046 2097 ! it matches. Set the KEYTYPES mask appropriately. This mask has
2047 2098 ! 16 bits, corresponding to standard datatype codes 0-15. (Codes
2048 2099 ! greater than 15 are not currently supported by Pascal for keys.)
2049 2100 ! If a bit for a particular datatype is set, then a key expression
2050 2101 ! of that type may be used to reference this key. Some combinations
2051 2102 ! require a range check of the key expression value. This is done
2052 2103 ! by PASS$FINDK. Then set the key size in KEYTYPES.
2053 2104 !-
2054 2105
2055 2106 The following table shows the allowable combinations of key value
```

```
2054 2105 2 | types and file key types. (ST1 refers to a 1-byte STG key)
2055 2106
2056 2107
2057 2108      STG  ST1  BN2  BN4  IN2  IN4  other
2058 2109      BU   no   (1)  ok   ok   no   no   no
2059 2110      WU   no   (1)  ok   ok   no   no   no
2060 2111      LU   no   (1)  (2)  ok   no   no   no
2061 2112      L    no   (1)  (2)  ck   (2)  ok   no
2062 2113      T,Z  ok   (3)  no   no   no   no   (4)
2063 2114
2064 2115 Notes:
2065 2116 1. Only if key not declared as string. Value must
2066 2117    be in range of key type.
2067 2118 2. Value must be in range of key type.
2068 2119 3. Only if key not declared as BU.
2069 2120 4. Can't be declared in key descriptor. Otherwise,
2070 2121    treated as string.
2071 2122
2072 2123 XAB_KEY = .XABKEY_ADDR [0]; ! Get first KEY XAB
2073 2124
2074 2125 INCR KEY_NUM FROM 0 TO .FCB [FCB$L_NKEYS] - 1 DO
2075 2126 BEGIN
2076 2127
2077 2128 LOCAL
2078 2129 KEY_DTYPE: BYTE, ! Datatype of key in DSC$ codes
2079 2130 USR_KEY: ! 1 if user defined key, otherwise 0
2080 2131
2081 2132 !+
2082 2133 ! Define literal masks for the datatypes.
2083 2134 !-
2084 2135 LITERAL
2085 2136 M_Z = 1^DSC$K_DTYPE_Z,
2086 2137 M_BU = 1^DSC$K_DTYPE_BU,
2087 2138 M_WU = 1^DSC$K_DTYPE_WU,
2088 2139 M_LU = 1^DSC$K_DTYPE_LU,
2089 2140 M_L = 1^DSC$K_DTYPE_L,
2090 2141 M_T = 1^DSC$K_DTYPE_T;
2091 2142
2092 2143 !+
2093 2144 ! Set USR_KEY if this key is in the KDB.
2094 2145 !-
2095 2146
2096 2147 USR_KEY = 0;
2097 2148 IF .KDB_NKEYS NEQ 0
2098 2149 THEN
2099 2150 IF .XAB_KEY [XAB$B_REF] EQL .KDB [KDB$B_KEY_NUMBER]
2100 2151 THEN
2101 2152 USR_KEY = 1;
2102 2153
2103 2154 !+
2104 2155 ! Find out what datatype the file's key is.
2105 2156 !-
2106 2157
2107 2158 CASE .XAB_KEY [XAB$B_DTP] FROM 0 TO XAB$C_MAXDTP OF
2108 2159 SET
2109 2160
2110 2161 [XAB$C_STG]: ! String
```

```
2111 BEGIN
2112 IF .USR_KEY
2113 THEN
2114 IF .KDB [KDB$B_DTYPE] EQL DSC$K_DTYPE_BU
2115 THEN
2116 BEGIN
2117 IF .XAB_KEY [XAB$B_TKS] NEQ 1
2118 THEN
2119 $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2120 KEYTYPES [0] = M_BU+M_WU+M_LU+M_L;
2121 END
2122 ELSE IF .KDB [KDB$B_DTYPE] EQL DSC$K_DTYPE_T
2123 THEN
2124 BEGIN
2125 IF .KDB [KDB$B_SIZE] NEQ .XAB_KEY [XAB$B_TKS]
2126 THEN
2127 $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2128 KEYTYPES [0] = M_T+M_Z;
2129 END
2130 ELSE
2131 $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM)
2132 ELSE
2133 IF .XAB_KEY [XAB$B_TKS] EQL 1
2134 THEN
2135 KEYTYPES [0] = M_BU+M_L+M_T+M_Z
2136 ELSE
2137 KEYTYPES [0] = M_T+M_Z;
2138 END;
2139 [XAB$C_IN2]: ! Word integer
2140 BEGIN
2141 IF .USR_KEY
2142 THEN
2143 IF .KDB [KDB$B_DTYPE] NEQ DSC$K_DTYPE_W
2144 THEN
2145 $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2146 KEYTYPES [0] = M_L;
2147 END;
2148 [XAB$C_IN4]: ! Longword integer
2149 BEGIN
2150 IF .USR_KEY
2151 THEN
2152 IF .KDB [KDB$B_DTYPE] NEQ DSC$K_DTYPE_L
2153 THEN
2154 $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2155 KEYTYPES [0] = M_L;
2156 END;
2157 [XAB$C_BN2]: ! Word unsigned
2158 BEGIN
2159 IF .USR_KEY
2160 THEN
2161 IF .KDB [KDB$B_DTYPE] NEQ DSC$K_DTYPE_WU
2162 THEN
2163 $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2164 KEYTYPES [0] = M_BU+M_WU+M_CU+M_L;
```



```
2168      END;
2169
2170      [XAB$C BN4]:      ! Longword unsigned
2171      BEGIN
2172      IF .USR_KEY
2173      THEN
2174      IF .KDB [KDB$B_DTYPE] NEQ DSC$K_DTYPE_LU
2175      THEN
2176      $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2177      KEYTYPES [0] = M_BU+M_WU+M_U+M_L;
2178      END;
2179
2180      [INRANGE]:      ! Defined by RMS, but not by PASCAL
2181      BEGIN
2182      +
2183      Pascal does not support this key type.
2184      However, if the file has one, it is treated as
2185      a string.
2186      -
2187      IF .USR_KEY
2188      THEN
2189      $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2190      KEYTYPES [0] = M_T+M_Z;
2191      END;
2192
2193      [OUTRANGE]:
2194      $PASSBUGCHECK (BUG_BADKEYDTP);
2195
2196      TES;
2197
2198      +
2199      Check offset of key.
2200      -
2201
2202      IF .USR_KEY
2203      THEN
2204      IF .KDB [KDB$L_OFFSET] NEQ .XAB_KEY [XAB$W_POS0]
2205      THEN
2206      $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2207
2208      +
2209      Set size of key.
2210      -
2211
2212      KEYTYPES [1] = .XAB_KEY [XAB$B_TKS];
2213
2214      +
2215      Advance KDB pointer, if any remain.
2216      -
2217
2218      IF .KDB_NKEYS GTR 0
2219      THEN
2220      BEGIN
2221      KDB = KDB [8,0,0,0];
2222      KDB_NKEYS = .KDB_NKEYS - 1;
2223      END;
2224
```

```
2225      2276      ! Advance XAB_KEY and KEYTYPES.
2226      2277      !
2227      2278      !
2228      2279      XAB_KEY = .XAB_KEY [XAB$NXT];
2229      2280      KEYTYPES = KEYTYPES [2];
2230      2281
2231      2282      END;      ! End of INCR Loop
2232      2283
2233      2284      !
2234      2285      ! If any KDB keys remain, give an error.
2235      2286      !
2236      2287      !
2237      2288      IF .KDB_NKEYS GTR 0
2238      2289      THEN
2239      2290      $PAS$IO_ERROR (PAS$KEYDEFINC,1,.KDB [KDB$B_KEY_NUMBER]);
2240      2291
2241      2292      RETURN;
2242      2293
2243      2294      ! End of routine CHECK_KEY_XABS
2244      2295      END;
```

```
.EXTRN SYSS$DISPLAY, PAS$K_KEYDEFINC
.EXTRN PAS$BUGCHECK
```

OF3C 00000 CHECK\_KEY\_XABS:

		5B	00000000G	00	9E	00002	WORD	Save R2,R3,R4,R5,R8,R9,R10,R11	1935
		5A	00000000G	00	9E	00009	MOVAB	PAS\$\$SIGNAL, R11	
FE		A7	40	8F	88	00010	MOVAB	PAS\$\$GET_VM, R10	
		54	04	AC	D0	00015	BISB2	#64, -2(FCB)	2001
		A7	09	A4	9A	00019	MOVL	XAB_SUM, R4	2007
DO		50	08	A6	D0	0001E	MOVZBL	9(R4), -48(FCB)	
				60	D5	00022	MOVL	8(PFV), PFD	2013
				0C	13	00024	TSTL	(PFD)	2014
	58	60		50	C1	00026	BEQL	1\$	
		59		88	9A	0002A	ADDL3	PFD, (PFD), KDB	2017
		58		03	C0	0002D	MOVZBL	(KDB)+, KDB_NKEYS	2018
				02	11	00030	ADDL2	#3, KDB	2019
				59	D4	00032	BRB	2\$	2014
				59	D1	00034	CLRL	KDB_NKEYS	2022
DO		A7		77	13	00038	CMPL	KDB_NKEYS, -48(FCB)	2031
		53	08	AC	D0	0003A	BEQL	8\$	
				63	D5	0003E	MOVL	XABKEY_ADDR, R3	2041
				0C	13	00040	TSTL	(R3)	
				53	DD	00042	BEQL	3\$	
			0C	BC	DD	00044	PUSHL	R3	2043
				02	FB	00047	PUSHL	@XABKEY_SIZE	
OC	BC	00000000G	00	8F	C5	0004E	CALLS	#2, PAS\$\$FREE_VM	
		DO	A7	8F	C5	0004E	MULL3	#76, -48(FCB), @XABKEY_SIZE	2049
				BC	DD	00058	PUSHL	@XABKEY_SIZE	2050
				01	FB	0005B	CALLS	#1, PAS\$\$GET_VM	
		6A		50	D0	0005E	MOVL	R0, XAB_KEY	
		52		52	D0	00061	MOVL	XAB_KEY, (R3)	2051
		63		54	D0	00064	MOVL	R4, LAST_XAB	2057
		50		01	CE	00067	MNEGL	#1, KEY_NUM	2059
		51		14	11	0006A	BRB	5\$	

		62	4C15	8F	B0	0006C	4\$:	MOVW	#19477, (XAB_KEY)	2061
	17	A2		51	90	00071		MOVW	KEY_NUM, 23(XAB_KEY)	2063
	04	A0		52	D0	00075		MOVL	XAB_KEY, 4(LAST_XAB)	2064
		50		82	7E	00079		MOVAB	(XAB_KEY)+, LAST_XAB	2065
		52	44	A2	9E	0007C		MOVAB	68(R2), XAB_KEY	2066
E7		51	D0	A7	F2	00080	5\$:	AOBLSS	-48(FCB), KEY_NUM, 4\$	2059
	68	A7		54	D0	00085		MOVL	R4, 104(FCB)	2074
		53	44	A7	9E	00089		MOVAB	68(R7), R3	2079
				53	DD	0008D	6\$:	PUSHL	R3	
00000000G		00		01	FB	0008F		CALLS	#1, SYS\$DISPLAY	
		18		50	E8	00096		BLBS	\$\$STATUS, 8\$	
0001825A		8F		50	D1	00099		CMPL	\$\$STATUS, #98906	
				04	12	000A0		BNEQ	7\$	
		E7	FF	A7	E8	000A2		BLBS	-1(FCB), 6\$	
		08		50	E8	000A6	7\$:	BLBS	\$\$STATUS, 8\$	
		7E	00G	8F	9A	000A9		MOVZBL	#PASSK_ERRDUROPE, -(SP)	2081
		68		01	FB	000AD		CALLS	#1, PASS\$SIGNAL	
					04	000B0		RET		
7E	D0	A7		02	78	000B1	8\$:	ASHL	#2, -48(FCB), -(SP)	2091
		6A		01	FB	000B6		CALLS	#1, PASS\$GET_VM	
		53		50	D0	000B9		MOVL	R0, KEYTYPES	
	CC	A7		53	D0	000BC		MOVL	KEYTYPES, -52(FCB)	2092
		52	08	BC	D0	000C0		MOVL	@XABKEY_ADDR, XAB_KEY	2123
		55		01	CE	000C4		MNEGL	#1, KEY_NUM	2125
			00BC	31	000C7			BRW	29\$	
				54	D4	000CA	9\$:	CLRL	USR_KEY	2147
				59	D5	000CC		TSTL	KDB_NKEYS	2148
				09	13	000CE		BEQL	10\$	
		68	17	A2	91	000D0		CMPB	23(XAB_KEY), (KDB)	2150
				03	12	000D4		BNEQ	10\$	
		54		01	D0	000D6		MOVL	#1, USR_KEY	2152
0052	07	00	13	A2	8F	000D9	10\$:	CASEB	19(XAB_KEY), #0, #7	2158
007B	0062	0047	001A			000DE	11\$:	.WORD	12\$-11\$,-	
	007B	007B	006B			000E6			16\$-11\$,-	
									20\$-11\$,-	
									18\$-11\$,-	
									21\$-11\$,-	
									23\$-11\$,-	
									23\$-11\$,-	
									23\$-11\$	
				05	DD	000EE		PUSHL	#5	2245
	00000000G	00		01	FB	000F0		CALLS	#1, PASS\$BUGCHECK	
					04	000F7		RET		
		1D		54	E9	000FB	12\$:	BLBC	USR_KEY, 15\$	2163
		02	01	A8	91	000FB		CMPB	1(KDB), #2	2165
				08	12	000FF		BNEQ	14\$	
		01	16	A2	91	00101		CMPB	22(XAB_KEY), #1	2168
				4B	13	00105	13\$:	BEQL	22\$	
				64	11	00107		BRB	26\$	2170
		0E	01	A8	91	00109	14\$:	CMPB	1(KDB), #14	2173
				5E	12	0010D		BNEQ	26\$	
	16	A2	02	A8	91	0010F		CMPB	2(KDB), 22(XAB_KEY)	2176
				46	13	00114		BEQL	24\$	
				55	11	00116		BRB	26\$	2178
		01	16	A2	91	00118	15\$:	CMPB	22(XAB_KEY), #1	2184
				3E	12	0011C		BNEQ	24\$	
		63	4105	8F	B0	0011E		MOVW	#16645, (KEYTYPES)	2186

PASSOPEN2  
1-015

OPEN procedure  
CHECK\_KEY\_XABS

- Check key XABs for indexed fil

H 15

16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 64  
(13)

			3C	11	00123		BRB	25\$		
	11		54	E9	00125	16\$:	BLBC	USR_KEY, 19\$		2193
	07	01	A8	91	00128		CMPB	1(KDB), #7		2195
			0B	13	0012C	17\$:	BEQL	19\$		
			3D	11	0012E		BRB	26\$		2197
	06		54	E9	00130	18\$:	BLBC	USR_KEY, 19\$		2203
	08	01	A8	91	00133		CMPB	1(KDB), #8		2205
			F3	11	00137		BRB	17\$		
	63	0100	8F	B0	00139	19\$:	MOVW	#256, (KEYTYPES)		2208
			21	11	0013E		BRB	25\$		2158
	0F		54	E9	00140	20\$:	BLBC	USR_KEY, 22\$		2213
	03	01	A8	91	00143		CMPB	1(KDB), #3		2215
			BC	11	00147		BRB	13\$		
	06		54	E9	00149	21\$:	BLBC	USR_KEY, 22\$		2223
	04	01	A8	91	0014C		CMPB	1(KDB), #4		2225
			B3	11	00150		BRB	13\$		
	63	011C	8F	B0	00152	22\$:	MOVW	#284, (KEYTYPES)		2228
			08	11	00157		BRB	25\$		2158
	11		54	E8	00159	23\$:	BLBS	USR_KEY, 26\$		2238
	63	4001	8F	B0	0015C	24\$:	MOVW	#16385, (KEYTYPES)		2241
	0D		54	E9	00161	25\$:	BLBC	USR_KEY, 27\$		2253
04	AB	1E	A2		00	ED	00164	CMPZV	#0, #16, 30(XAB_KEY), 4(KDB)	2255
			04	13	0016B		BEQL	27\$		
			55	DD	0016D	26\$:	PUSHL	KEY_NUM		2257
			26	11	0016F		BRB	32\$		
	02	A3	16	A2	9B	00171	27\$:	MOVZBW	22(XAB_KEY), 2(KEYTYPES)	2263
				59	D5	00176		TSTL	KDB_NKEYS	2269
				05	15	00178		BLEQ	28\$	
	58			08	C0	0017A		ADDL2	#8, KDB	2272
				59	D7	0017D		DECL	KDB_NKEYS	2273
	52	04	A2	D0	0017F	28\$:	MOVL	4(XAB_KEY), XAB_KEY		2280
	53		04	C0	00183		ADDL2	#4, KEYTYPES		2281
	55	D0	A7	F2	00186	29\$:	AOBLS	-48(FCB), KEY_NUM, 30\$		2125
			03	11	0018B		BRB	31\$		
			FF3A	31	0018D	30\$:	BRW	9\$		
			59	D5	00190	31\$:	TSTL	KDB_NKEYS		2289
			0C	15	00192		BLEQ	33\$		
	7E		68	9A	00194		MOVZBL	(KDB), -(SP)		2291
			01	DD	00197	32\$:	PUSHL	#1		
	7E	00G	8F	9A	00199		MOVZBL	#PASSK KEYDEF INC, -(SP)		
	6B		03	FB	0019D		CALLS	#3, PASS\$SIGNAL		
			04	001A0	33\$:		RET			2295

; Routine Size: 417 bytes, Routine Base: \_PASSCODE + 0B54



```
2246 2296 1 %SBTTL 'OPEN_HANDLER - Close file on unwind'
2247 2297 1 ROUTINE OPEN_HANDLER (
2248 2298 1     SIGNAL_ARGS: REF BLOCK [, BYTE]          ! Signal arguments
2249 2299 1     MECHANISM_ARGS: REF BLOCK [, BYTE],      ! Mechanism arguments
2250 2300 1     ENABLE_ARGS: REF VECTOR [, LONG]      ! Enable arguments
2251 2301 1 ) =
2252 2302 1
2253 2303 1
2254 2304 1 ++
2255 2305 1 FUNCTIONAL DESCRIPTION:
2256 2306 1     OPEN_HANDLER is a condition handler established by PAS$OPEN. It
2257 2307 1     serves only to close the file which is currently being opened
2258 2308 1     if an unwind occurs.
2259 2309 1
2260 2310 1 CALLING SEQUENCE:
2261 2311 1
2262 2312 1     ret_status.wlc.v = OPEN_HANDLER (signal_args.rlu.r, mechanism_args.rlu.r,
2263 2313 1     enable_args.rlu.r)
2264 2314 1
2265 2315 1 FORMAL PARAMETERS:
2266 2316 1
2267 2317 1     SIGNAL_ARGS
2268 2318 1     MECHANISM_ARGS
2269 2319 1     ENABLE_ARGS
2270 2320 1
2271 2321 1     - The signal arguments list
2272 2322 1     - The mechanism argument list
2273 2323 1     - A vector of four longwords, the first of
2274 2324 1       which contains the count of remaining
2275 2325 1       longwords (3). The remaining longwords
2276 2326 1       contain the following information:
2277 2327 1       ENABLE_ARGS [1] - Address of longword which
2278 2328 1       contains the address of the
2279 2329 1       active PFV.
2280 2330 1       ENABLE_ARGS [2] - Address of longword which
2281 2331 1       contains the address of
2282 2332 1       a block of KEY XABs to be
2283 2333 1       deallocated (if non-zero).
2284 2334 1       ENABLE_ARGS [3] - Address of longword which
2285 2335 1       contains the number of bytes
2286 2336 1       of KEY XABs to deallocate.
2287 2337 1
2288 2338 1 IMPLICIT INPUTS:
2289 2339 1
2290 2340 1     NONE
2291 2341 1
2292 2342 1 IMPLICIT OUTPUTS:
2293 2343 1
2294 2344 1     NONE
2295 2345 1
2296 2346 1 COMPLETION STATUS:
2297 2347 1
2298 2348 1     SSS_RESIGNAL
2299 2349 1
2300 2350 1 SIDE EFFECTS:
2301 2351 1
2302 2352 1     Closes the current file upon an unwind.
2303 2353 1     Deallocates KEY XABs.
2304 2354 1
2305 2355 1 SIGNALLED ERRORS:
```

```
2303 2353 1 NONE
2304 2354 1
2305 2355 1
2306 2356 1
2307 2357 2 BEGIN
2308 2358 2
2309 2359 2
2310 2360 2
2311 2361 2
2312 2362 2
2313 2363 2 IF .SIGNAL_ARGS [CHF&L_SIG_NAME] EQL SS&_UNWIND
2314 2364 2 THEN
2315 2365 2 BEGIN
2316 2366 2
2317 2367 2 LOCAL
2318 2368 2 PFV: REF $PASSPFV_FILE_VARIABLE; ! Pascal File Variable
2319 2369 2
2320 2370 2
2321 2371 2
2322 2372 2
2323 2373 2
2324 2374 2
2325 2375 2 PFV = ..ENABLE_ARGS [1];
2326 2376 2 IF PFV [PFV$R_PFV] NEQA 0
2327 2377 2 THEN
2328 2378 2 IF .PFV [PFV$V_FCB_VALID]
2329 2379 2 THEN
2330 2380 2 BEGIN
2331 2381 2 PASS$CLOSE (PFV [PFV$R_PFV]); ! Close the file
2332 2382 2 PASS$REMOVE_FILE (.PFV [PFV$A_FCB]); ! Deallocate FCB
2333 2383 2 END;
2334 2384 2
2335 2385 2
2336 2386 2
2337 2387 2
2338 2388 2
2339 2389 2 IF ..ENABLE_ARGS [2] NEQ 0
2340 2390 2 THEN
2341 2391 2 PASS$FREE_VM (..ENABLE_ARGS [3], .ENABLE_ARGS [2]);
2342 2392 2 END;
2343 2393 2
2344 2394 2 RETURN SS&_RESIGNAL;
2345 2395 2
2346 2396 2 END; ! End of routine OPEN_HANDLER
```

.EXTRN PASS\$CLOSE, PASS\$REMOVE\_FILE

		000C 0000 OPEN_HANDLER:						
					.WORD	Save R2,R3	2297	
00000920	50	04	AC	D0	00002	MOVL	SIGNAL_ARGS, R0	2363
	8F	04	A0	D1	00006	CMPL	4(R0), #2336	
			34	12	0000E	BNEQ	2\$	
	52	0C	AC	D0	00010	MOVL	ENABLE_ARGS, R2	2375
	53	04	B2	D0	00014	MOVL	24(R2), PFV	
			18	13	00018	BEQL	1\$	2376

PASSOPEN2  
1-015

OPEN procedure  
OPEN\_HANDLER - Close file on unwind

K 15  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 67  
(14)

13	07	A3	06	E1	0001A	BBC	#6, 7(PFV), 1\$	: 2378
			53	DD	0001F	PUSHL	PFV	: 2381
00000000G	00		01	FB	00021	CALLS	#1, PASS\$CLOSE	
		0C	A3	DD	00028	PUSHL	12(PFV)	: 2382
00000000G	00		01	FB	0002B	CALLS	#1, PASS\$REMOVE_FILE	
		08	B2	D5	00032	TSTL	@8(R2)	: 2389
			0D	13	00035	BEQL	2\$	
		08	A2	DD	00037	PUSHL	8(R2)	: 2391
		0C	B2	DD	0003A	PUSHL	@12(R2)	
00000000G	00		02	FB	0003D	CALLS	#2, PASS\$FREE_VM	
	50	0918	8F	3C	00044	MOVZWL	#2328, R0	: 2394
			04	00049	RET			: 2396

; Routine Size: 74 bytes, Routine Base: \_PASS\$CODE + 0CF5

: 2347  
: 2348  
2397 1  
2398 1 !<BLF/PAGE>

```
: 2350      2399 1 %SBTTL 'EXIT_HANDLER - Exit handler for file system'
: 2351      2400 1 ROUTINE EXIT_HANDLER (
: 2352      2401 1     EXIT_REASON: REF VECTOR [, LONG]           ! Exit reason
: 2353      2402 1     ): NOVALUE =
: 2354      2403 1
: 2355      2404 1 ++
: 2356      2405 1 FUNCTIONAL DESCRIPTION:
: 2357      2406 1
: 2358      2407 1     This is the exit handler for the file system. It is declared
: 2359      2408 1     by PAS$OPEN, and serves to close all open files in an orderly
: 2360      2409 1     fashion upon image exit.
: 2361      2410 1
: 2362      2411 1 CALLING SEQUENCE:
: 2363      2412 1
: 2364      2413 1     CALL EXIT_HANDLER (EXIT_REASON.rlc.r)
: 2365      2414 1     (Called by VMS upon image exit.)
: 2366      2415 1
: 2367      2416 1 FORMAL PARAMETERS:
: 2368      2417 1
: 2369      2418 1     EXIT_REASON      - The reason for the exit. This parameter is
: 2370      2419 1     not used here.
: 2371      2420 1
: 2372      2421 1 IMPLICIT INPUTS:
: 2373      2422 1
: 2374      2423 1     NONE
: 2375      2424 1
: 2376      2425 1 IMPLICIT OUTPUTS:
: 2377      2426 1
: 2378      2427 1     NONE
: 2379      2428 1
: 2380      2429 1 COMPLETION STATUS:
: 2381      2430 1
: 2382      2431 1     NONE
: 2383      2432 1
: 2384      2433 1 SIDE EFFECTS:
: 2385      2434 1
: 2386      2435 1     Closes all open files.
: 2387      2436 1
: 2388      2437 1 SIGNALLED ERRORS:
: 2389      2438 1
: 2390      2439 1     NONE
: 2391      2440 1
: 2392      2441 1 --
: 2393      2442 1
: 2394      2443 2 BEGIN
: 2395      2444 2
: 2396      2445 2 !+
: 2397      2446 2 ! Clear EXITH_DECLARED so that if a user exit handler opens more
: 2398      2447 2 ! files, another handler will be declared.
: 2399      2448 2 !-
: 2400      2449 2
: 2401      2450 2 EXITH_DECLARED = 0;
: 2402      2451 2
: 2403      2452 2 !+
: 2404      2453 2 ! Call PAS$$CLOSE_ALL to close all the files.
: 2405      2454 2 !-
: 2406      2455 2
```



PASSOPEN2  
1-015

OPEN procedure  
EXIT\_HANDLER - Exit handler for file system

M 15  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 69  
(15)

: 2407      2456   2      PASS\$CLOSE\_ALL ();  
: 2408      2457   2  
: 2409      2458   2      RETURN;  
: 2410      2459   2  
: 2411      2460   1      END;

! End of routine EXIT\_HANDLER

.EXTRN PASS\$CLOSE\_ALL

0000 00000 EXIT\_HANDLER:

00000000G 00 00000000' EF D4 00002  
00 FB 00008  
04 0000F

.WORD Save nothing  
CLRL EXITH DECLARED  
CALLS #0, PASS\$CLOSE\_ALL  
RET

: 2400  
: 2450  
: 2456  
: 2460

: Routine Size: 16 bytes,      Routine Base: \_PASS\$CODE + 0D3F

: 2412      2461   1  
: 2413      2462   1 !<BLF/PAGE>

PASSOPEN2  
1-015

OPEN procedure  
EXIT\_HANDLER - Exit handler for file system

N 15  
16-Sep-1984 01:46:15  
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742  
[PASRTL.SRC]PASOPEN2.B32;1

Page 70  
(16)

: 2415 2463 1 END  
: 2416 2464 1  
: 2417 2465 0 ELUDOM

! End of module PASSOPEN2

# PSECT SUMMARY

Name	Bytes	Attributes
PASSDATA	6	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
PASSCODE	3407	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

# Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	151	1	581	00:01.0
\$255\$DUA28:[PASRTL.OBJ]PASLIB.L32;1	427	195	45	33	00:00.4

# COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:PASOPEN2/OBJ=OBJ\$:PASOPEN2 MSRC\$:PASOPEN2/UPDATE=(ENH\$:PASOPEN2)

Size: 3298 code + 115 data bytes  
Run Time: 01:27.8  
Elapsed Time: 04:18.1  
Lines/CPU Min: 1684  
Lexemes/CPU-Min: 21607  
Memory Used: 824 pages  
Compilation Complete



0295 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY